175 PTAS



CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen IX-Fascículo 100

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Aribau, 185, 1.°, 08021 Barcelona
Tel. (93) 209 80 22 - Télex: 93392 EPPA

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-181-X (tomo 9)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 188512
Impreso en España-Printed in Spain-Diciembre 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S. A. (Aribau, 185, 1.°, 08021 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

No se efectúan envíos contra reembolso.



Futuro luminoso

Acerquémonos a la informática óptica, tecnología que probablemente hará parecer lentas las más veloces máquinas de hoy en día





Fibra añadida
El equivalente óptico del cable eléctrico es el cable de fibra óptica. Este cable permite «doblar» las señales de luz y, en consecuencia, dirigirlas hacia donde deseemos. Las fibras de que está relleno el cable contienen una sustancia transparente que «conduce» la luz en la dirección de la fibra

Science Photo Library Ltd.'

Cuando se piensa en un ordenador, casi con seguridad se imagina un dispositivo enchufable, basado en interruptores binarios electrónicos que almacenan y procesan la información. Sin embargo, usted estaría equivocado si pensara que éste es el único tipo viable de ordenador. Se podría concebir un ordenador basado en cualquier método para el almacenamiento de la información de forma física: una carga eléctrica, los dientes y las muescas de las ruedas de engranaje, orificios en tiras de papel, cuentas, piedrecillas, conchas marinas o cualquier otro medio.

El semiconductor electrónico es el mejor de los tipos de almacenamiento de información que se ha inventado hasta ahora, porque su consumo de potencia es ínfimo, su tamaño es microscópico y, lo más importante, es rápido. La velocidad a la que puede operar un ordenador está limitada por el tiempo que requieren sus almacenamientos de información binarios, o *interruptores*, para pasar de una posición a otra, y por la velocidad de la transmisión de información entre interruptores. Mientras que los circuitos de ordenador basados en corrientes de electrones son muy veloces para realizar estas funciones, teórica y técnicamente es posible construir circuitos aún más rápidos, basados en haces de luz.

La investigación en el campo de la informática óptica y disciplinas afines, como la *lógica fotónica*, ya se ha puesto en marcha. Muchos de los componentes que se incorporarían en un ordenador óptico ya están disponibles y en uso. Las fibras ópticas, por ejemplo, se están utilizando para transportar señales telefónicas y de televisión. Se han desarrollado guías de onda ópticas, equivalentes a las pistas metálicas de las placas de circuito impreso. También se han desarrollado discos ópticos. El problema más crucial de todos, la analogía óptica del interruptor binario electrónico, está actualmente en fase de prueba.

Los diseños para un interruptor binario óptico se agrupan en dos tipos principales: los interruptores electroópticos y los interruptores totalmente ópticos. Es interesante analizar cada uno de los dos tipos, empezando por el híbrido electroóptico, puesto que los componentes ópticos probablemente se utilizarán primero en conjunción con componentes electrónicos, en lugar de entrar a competir con ellos. Ambos tipos de interruptores explotan el fenómeno de la interferencia de luz provocada por cambios controlados del índice de refracción de un material óptico no lineal, por lo general un compuesto de galio o litio. El índice de refracción de un material expresa la relación de la velocidad de la luz en el vacío con su velocidad en el material. Por ejemplo, si el índice de refracción de un tipo deter-





Science Photo Library Ltd.

El programa de luz
Otro importante componente que será necesario para hacer realidad el ordenador óptico es el dispositivo semiconductor, que se puede regular para que emita una gama de frecuencias de luz pura. Éste será vital a los fines de interferencias en la comunicación, donde se requiere luz a una frecuencia y longitud de onda estándares

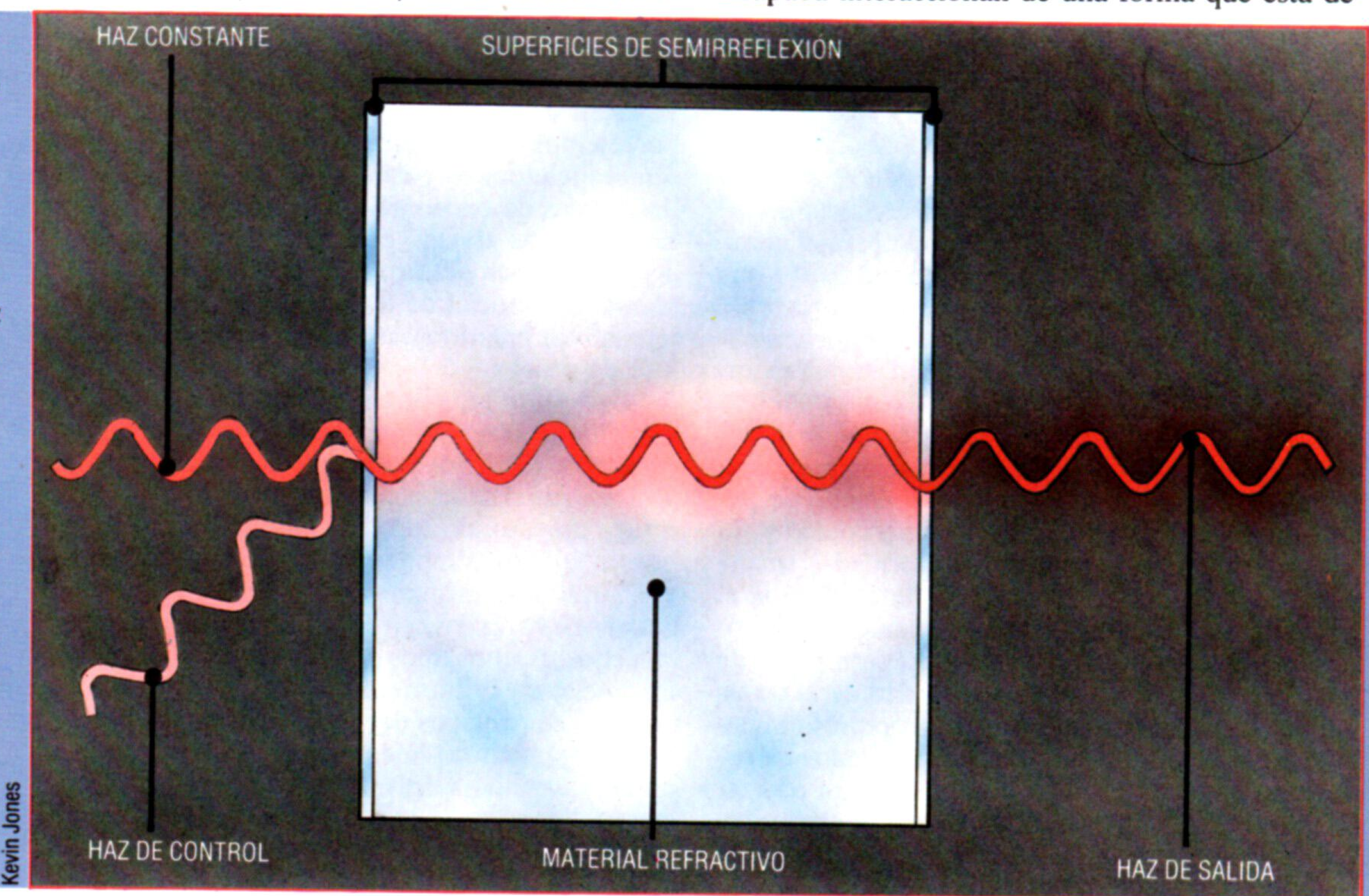
minado de vidrio es 1,5, luego, como la luz viaja a alrededor de 300 000 km/s en el vacío, su velocidad será de alrededor de 200 000 km/s en el vidrio.

El tipo de interruptor binario híbrido electroóptico se basa en un invento denominado *interferómetro Mach Zehnder*. El principio de este dispositivo es muy simple. Se divide en dos un haz de luz entrante, se le conduce a través de canales separados de material no lineal y después se lo vuelve a recombinar. Si no se aplica ninguna corriente eléctrica a ninguno de los canales del interferómetro, los haces separados llegan a la salida en fase y generan un haz intenso (el estado *on*). Si se altera el índice de refracción de alguno de los canales mediante una señal eléctrica, los dos haces ya no se recombinan en fase, puesto que uno de los dos haces viajará más rápidamente que el otro, y el resultado es un haz de salida mucho más débil (el estado *off*).

Este tipo de interruptor podría conformar la base de un circuito integrado óptico sencillo. El circuito sería más veloz que un equivalente electrónico debido a la capacidad de la luz de transportar la información a mayor velocidad que las señales eléctricas (alrededor de 10 000 veces más rápido, debido a la mayor frecuencia de las ondas de luz). A pesar de este rendimiento superior, la velocidad de este circuito híbrido aún es limitada, ya que para la conmutación se basa en la realimentación eléctrica. Mediante el empleo de luz para operar el interruptor (utilizando un haz más débil para controlar uno más intenso) se pueden obtener velocidades superiores.

Este tipo de diseño es, por supuesto, exactamente análogo al transistor electrónico. Un equipo de la Heriot Watt University, de Edimburgo, ya ha construido una versión prototipo de interruptor totalmente óptico, a la que se ha denominado *transfasor*. El dispositivo realiza la conmutación en alrededor de un picosegundo (una milbillonésima de segundo). El transistor electrónico más veloz, por su parte, posee una velocidad de conmutación de un nanosegundo (una billonésima de segundo). Esto significa que un ordenador totalmente óptico podría operar potencialmente a una velocidad mil veces superior a la de un ordenador electrónico.

El transfasor desarrollado en Heriot Watt se basa en el diseño de un interferómetro inventado por los físicos franceses Charles Fabry y Alfred Perot en 1896. Entre dos superficies reflectantes, se intercala una fina lámina de material no lineal. La luz que entra por una superficie rebota repetidamente dentro de la capa intercalada antes de salir a través de la superficie opuesta. Las ondas de luz atrapada interaccionan de una forma que está de-



Kevin Jones

En el haz
El bloque de construcción básico del ordenador electrónico es el transistor, cuyo equivalente óptico se denomina *transfasor*. Por uno de los extremos se introduce un haz constante junto con un «haz de control», que se interferirán entre sí en el interior del dispositivo. Una pequeña variación en la intensidad del haz de control producirá un gran cambio en la intensidad de la luz que salga. De este modo el transfasor se puede utilizar como un dispositivo de conmutación lógico

terminada por el índice de refracción del material no lineal, que se puede regular utilizando un haz de control. Si el índice de refracción es tal que las ondas de luz están en fase y se refuerzan entre sí, entonces se transmite un haz de luz intenso. Si las ondas están desfasadas, sólo se transmite un haz de luz débil. Los dos estados de transmisión alternativos corresponden a las posiciones *on* (encendido) y *off* (apagado).

Beneficios teóricos

El éxito del ordenador óptico, en definitiva, estará determinado por los límites teóricos y técnicos de las máquinas convencionales. Sobre el papel, los ordenadores ópticos son superiores a sus homónimos electrónicos en numerosos sentidos. El primero de ellos es la velocidad de conmutación. Para que un transistor actúe como interruptor, una corriente de electrones debe atravesar su base. La velocidad a la cual pueden desplazarse los electrones en un semiconductor tiene límites, así como el espesor con que se puede construir un transistor (por supuesto, cuanto más fina sea la base, tanto más corto será el viaje de los electrones). Un ordenador óptico, transmitiendo a la velocidad de la luz, es intrínsecamente mucho más veloz.

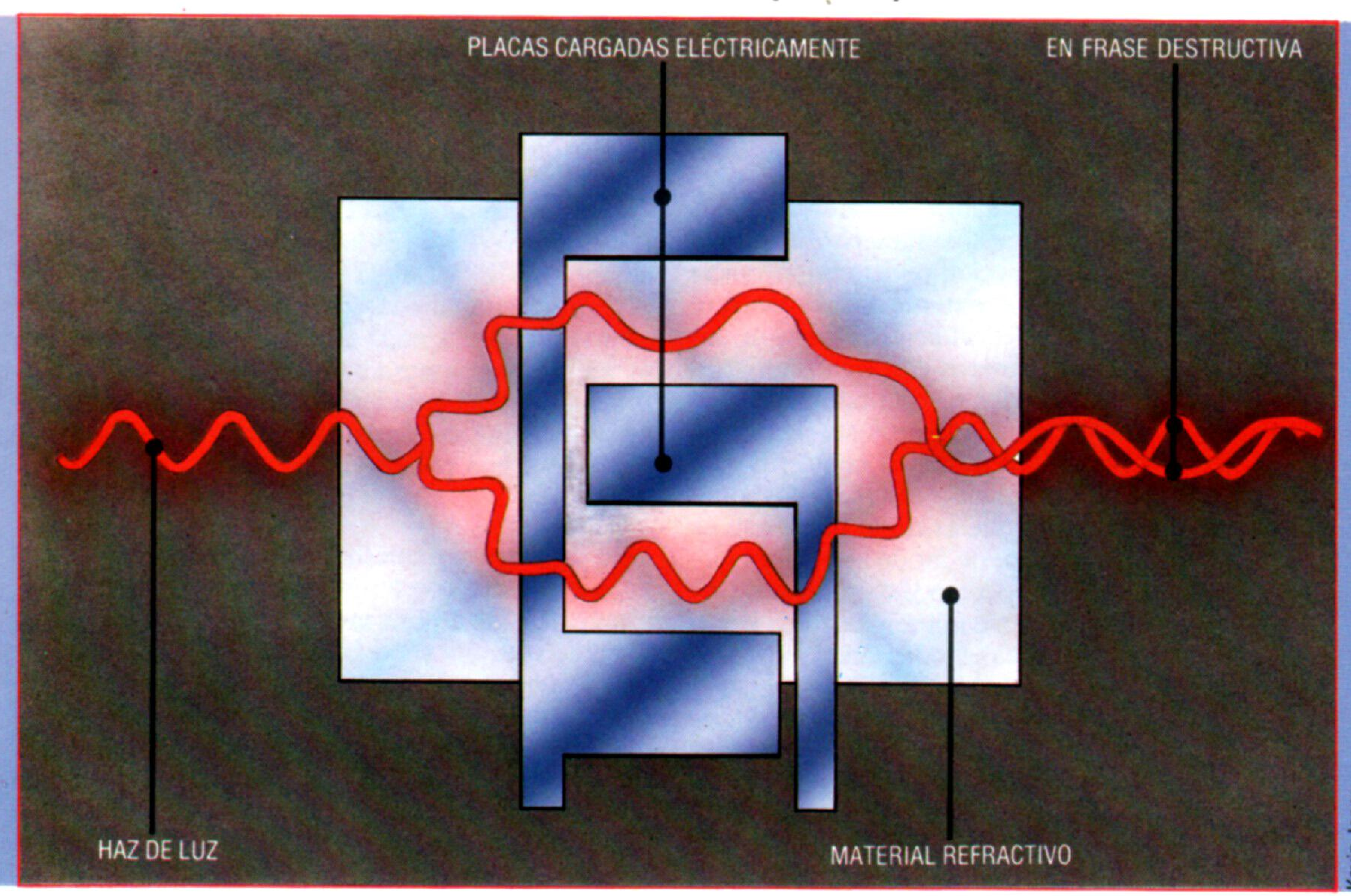
Las máquinas convencionales también están limitadas por lo que se denomina *el cuello de botella de Von Neumann*. Ésta es una característica básica de casi todos los ordenadores y está relacionada con la forma en que se accede a la información desde la memoria a razón de una palabra por vez (estilo «en serie»). En un ordenador óptico, se podría acceder a la memoria en paralelo. Ello se debe a que no es necesario aislar los haces de luz igual que las señales eléctricas. Por lo tanto, se podrían canalizar haces separados a través del mismo transistor óptico y conmutarlos simultáneamente.

La tercera limitación del diseño del ordenador convencional es el limitado espacio disponible para realizar conexiones en un chip plano. Por lo general, la cantidad de conexiones que se pueden efectuar en un solo chip está severamente limitada por el problema de ubicación de patillas, un límite sobre la cantidad de patillas que se pueden montar sobre un trozo plano de silicio. Aplicando técnicas holográficas (ópticas) tridimensionales, en principio sería posible introducir un grado de libertad adicional para la disposición de las conexiones, y cambiar instantáneamente su configuración.

Por último, están los beneficios prácticos de la interconexión entre los ordenadores ópticos y los periféricos. Ya se están utilizando redes de fibras ópticas para enlazar dispositivos electrónicos. Son sumamente rápidas y en la misma fibra óptica se pueden transmitir mensajes de luz sin ninguna interferencia. Los ordenadores ópticos se podrían conectar entre sí mediante estas redes sin necesidad de ninguna interface optoeléctrica lenta.

A la vista de estas ventajas, ¿cuándo podemos esperar que comiencen a aparecer ordenadores ópticos prácticos y asequibles? La respuesta es: no aparecerán por ahora. Los beneficios teóricos de la óptica en comparación con la electrónica no son los mismos en la práctica. Los impresionantes avances en la miniaturización del chip significan que, al menos a corto plazo, los ordenadores ópticos representan un objetivo que retrocede constantemente. La cantidad de dispositivos de transistor que se pueden empaquetar en un chip continúa duplicándose cada dieciocho meses. Para el año 2000, las predicciones hablan de chips con 10 o 100 millones de dispositivos. La integración a gran escala (LSI: *large scale integration*) ha pasado a ser una integración a muy grande (y ultra) escala. El «procesador en paralelo» se ha convertido en el «procesador paralelo en masa».

La última tecnología electrónica de gran velocidad está llamada a seguir perfeccionándose y, con esta clase de competencia, puede ser que el ordenador óptico asequible se halle todavía a años luz.



La fase
El interferómetro Mach Zehnder posibilita el trabajo conjunto de dispositivos electrónicos y ópticos. La luz que entra por un extremo se divide en dos haces. Una vez que los haces han entrado en el dispositivo (regulando el índice de refracción para uno de los haces), la velocidad de la luz se reducirá y quedará «desfasada» con relación al otro haz. Cuando los dos haces converjan en el haz de salida, las fases se interferirán destructivamente y el resultado será *off*, que se puede considerar como un cero lógico

Criba de números

Las habilidades aritméticas del FORTH se ven realzadas en el programa «La criba de Eratóstenes»

Una de las características del FORTH que puede resultar enigmática al principio es su restricción a la aritmética de enteros. Esto es algo que no viene dado por razones de interactividad y ampliabilidad, sino por razones de eficiencia: la aritmética de enteros es mucho más veloz que la de punto flotante.

En FORTH, la forma habitual de manipular fracciones es con aritmética de *punto fijo*, en la que se emplean números que se multiplican o dividen por alguna constante, por lo general una potencia de diez. Esto es como usar milímetros cuando los metros no son suficientemente exactos.

Un número de la pila del FORTH posee 16 bits, y, por lo tanto, puede considerarse como un entero *con signo* (indicándose el signo mediante el bit más significativo) entre −32768 y −32767, o un entero *sin signo* entre 0 y 65535. La forma más rápida de ver esta ambigüedad es digitando:

```
65535.
```

Esto visualiza la respuesta −1, porque el . trata a la parte superior de la pila con un número con signo, y el número sin signo 65535 y el número con signo −1 se representan mediante la misma entrada en la pila. La palabra del FORTH U. es igual que ., con la excepción de que trata a la parte superior de la pila como si no tuviera signo. De modo que:

```
65535 U.
```

visualiza 65535.

Existen algunas otras palabras en diferentes versiones, según esperen que los números de la pila tengan o no signo. La elección reviste importancia si existe la posibilidad de obtener números entre 32768 y 65535. Hay que decidir el empleo de las palabras, ya sea para números con signo, en cuyo caso estos números grandes se cuentan como desbordamiento, o bien para números sin signo, en cuyo caso no se permiten números negativos. Un buen ejemplo es cuando se emplean números como direcciones de memoria (que es como funcionan @ y !). Éstos no tienen signo, de modo que habrá que utilizar palabras sin signo, como U< en lugar de <. Asimismo, pueden utilizarse números de pila como patrones de 16 bits y realizar las combinaciones booleanas entre dos bits habituales.

El ejemplo principal muestra un programa que utiliza la «criba de Eratóstenes» para visualizar todos los primos hasta el 65536. Como es común en los programas en FORTH, resultará más fácil leerlo hacia atrás: empezar por la palabra principal, PRIMOS, e ir hacia atrás para hallar las definiciones de las palabras empleadas. PRIMOS es la palabra a escribir primero, pero deberá digitarse al final, después de sus subpalabras. Para el programa *La criba de Eratóstenes*, se van escribiendo todos los números hasta un límite especificado. Al encontrar un primo, se tachan todos sus múltiplos, que obviamente no pueden ser también primos, y el siguiente primo es el siguiente número que no se ha tachado. Ignorando el 1 (que, por diversas razones, no se considera un auténtico primo), se encuentra el primer primo, 2, y se tachan todos sus múltiplos. El siguiente número que no se ha tachado, el 3, es el siguiente primo, de modo que se tachan todos sus múltiplos y así sucesivamente. Se hace muy fácil con lápiz y papel hasta hallar todos los primos hasta 20.

El programa en FORTH listado aquí almacena ocho Kbytes o 65536 bits, uno para cada número hasta 65536. Al comienzo todos ellos están en 0, pero al tachar uno, su bit se cambia a 1. El espacio de memoria para estos bytes se prepara mediante:

```
CREATE BITS 8192 ALLOT
```

(En realidad hemos reemplazado 8192 por una constante, BYTES.) Más adelante veremos con más precisión cómo trabaja esto, pero por ahora su efecto consiste en asignar un bloque de 8192 bytes a algún lugar de la memoria y crear una nueva palabra, BITS, cuya acción consiste en apilar la dirección del primero de estos bytes. Se puede utilizar, por lo tanto, para que contenga una matriz de bytes.

Para referirse a uno de los bits, se necesitan dos números: uno que muestre qué byte contiene el bit (de modo que para este número pueda utilizarse la dirección de memoria del byte) y otro que muestre qué bit se halla dentro de ese byte, de modo que estará entre 0 y 7. No es demasiado difícil deducir de este par de números el número (entre 1 y 65536) que representa el bit.

Recuérdese que CURBYTES es una variable; su valor, CURBYTES@, es una dirección de un byte de la matriz BITS, y CURBYTES @ C @ es el valor del byte. Si se está familiarizado con el lenguaje C, esta idea no resultará nada extraña. Dado que los primos y las direcciones no tienen signo y posiblemente sean bastante grandes, ocasionalmente se necesitarán palabras sin signo como U. y U<.

He aquí una palabra útil:

```
+! (n, dirección –)
```

Ésta sustituye el número de la dirección dada por el mismo número con n sumado, de modo que típicamente se utiliza para incrementar el valor de una variable con una cantidad dada.

Un punto final acerca de la aritmética es que algunas palabras son para aritmética de *doble longitud*. Esto significa que se utilizan juntos dos números de dos bytes de la pila para formar un único número de cuatro bytes. La mitad más significativa es la más próxima a la parte superior de la pila, y la mitad menos significativa la de abajo.

Bases numéricas
En FORTH es muy fácil cambiar de base numérica, de modo que los números se impriman o se lean desde el teclado como binarios, octales, decimales, hexadecimales o en cualquier otra base numérica que se escoja. Esto se consigue cambiando la variable BASE. P. ej.:

```
54 16 BASE !. DECIMAL
```

Esto visualiza 36 (54 en hexa), vuelve a cambiar la base numérica a diez con la palabra DECIMAL.

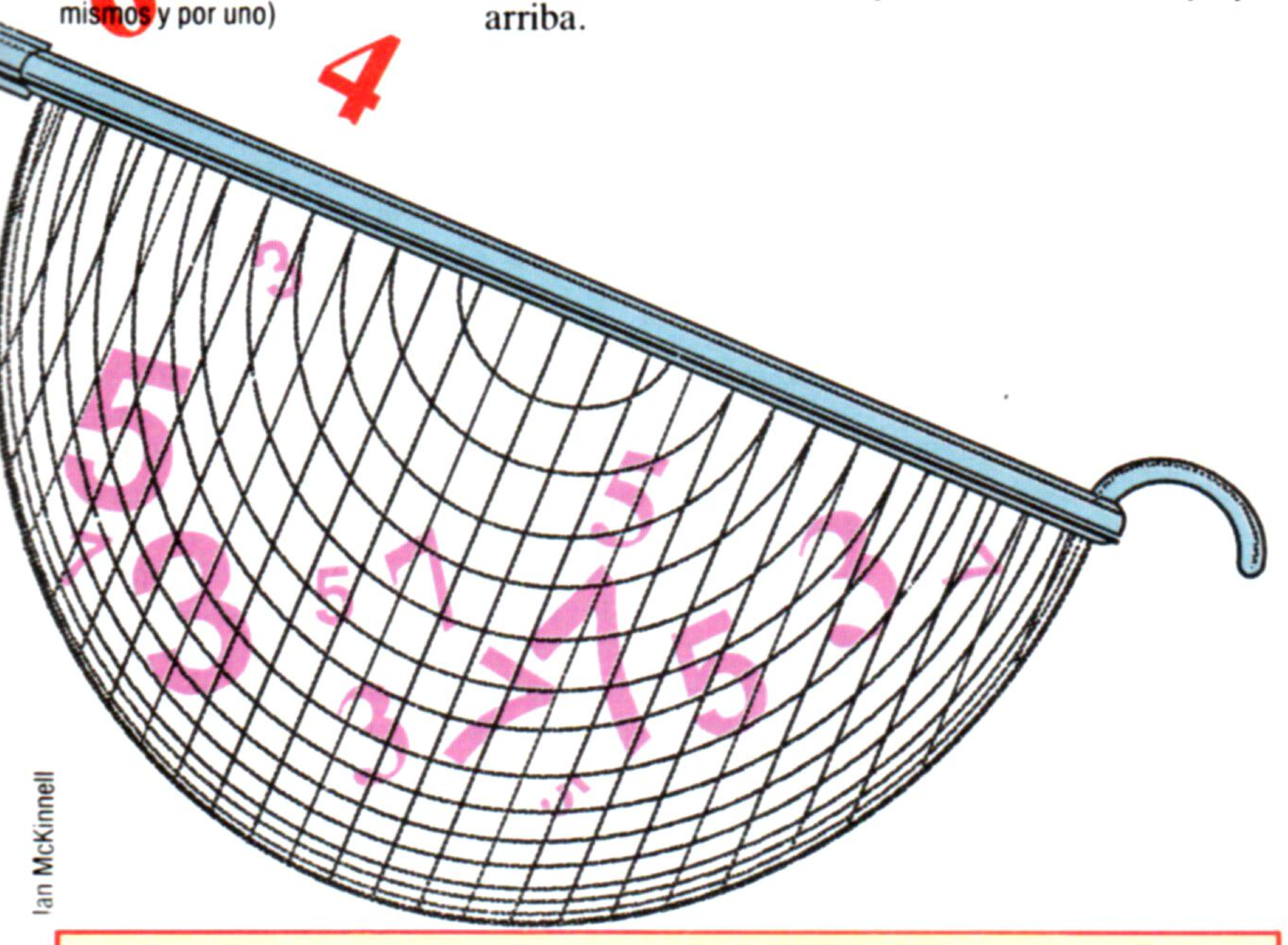
En la práctica resultará más sencillo definir palabras como

```
:HEX 16 BASE !;
```

Obsérvese que una vez que un número (como 16 aquí) forma parte de la definición de una palabra, no se ve afectado por posteriores cambios de la base numérica



Selección de primos
La criba de Eratóstenes, así llamada en honor del matemático griego que la formuló, es un método simple de hallar números primos. El método trabaja «tamizando» múltiplos de números sucesivos hasta que sólo quedan números primos (divisibles sólo por sí mismos y por uno)



Ian McKinnell

Aunque el estándar FORTH-83 sólo permite digitar números de longitud simple, en la mayoría de las implementaciones, si un número contiene un punto (en cualquier lugar dentro del número), se ignora (no se trata como un punto decimal) y el número se coloca en la pila como uno de doble longitud. Por ejemplo, 10.000 es 10000 de doble longitud: dos entradas de la pila, con 10000 abajo y 0 arriba.

```
8192 CONSTANT BYTES
CREATE BITS BYTES ALLOT BITS BYTES
+ CONSTANT BITSEND
(   1 bit para cada número 1 – 65536 )
VARIABLE CURBYTE VARIABLE CURBIT
(   mostrar número que se está examinando.
CURBYTE   )
(   señala byte de BITS, CURBIT a bit )
VARIABLE MASK ( 2**CURBIT )
VARIABLE PRIMO VARIABLE PRIMO/8
VARIABLE PRIMOMOD8
( mostrar primo una vez hallado )
: SETUP ( – – ) ( inicializa variables )
  BYTES 0 DO ( BITS de matriz cero )
    0 BITS I + C!
  LOOP
  BITS CURBYTE ! 0 CURBIT ! 1 MASK!
;
: BITSIG ( – – 0 o no – cero )
: regula CURBYTE etc para hallar siguiente bit )
( y lo coloca en la pila   )
  1 CURBIT + !
  MASK @ 2 * MASK !
  CURBIT @ 8 = IF
    0 CURBIT !
    1 MASK !
    1 CURBYTE + !
  THEN
  CURBYTE @ C@ MASK @ AND
;
: PRIMOETC (    – –    ) (   prepara PRIMO,
  PRIMO/8   )
( y PRIMOMOD8   )
  CURBIT @ 1 +
  DUP 8 = IF
    DROP 0 PRIMOMOD8 ! 1
  ELSE
    PRIMOMOD8 ! 0
  THEN
  CURBYTE @ BITS – + PRIMO/8 !
  PRIMO/8 @ 8 * PRIMOMOD8 @ + PRIMO !
;
: OTRO-PRIMO (    – – 0 o –1   )
( regula CURBYTE etc. al siguiente primo,
  establece   )
( PRIMO etc. deja 0 si se ha llegado al final de
  BITS   )
  BEGIN
  BITSIG 0= UNTIL
  CURBYTE @ BITSEND U< IF
    PRIMOETC
    –1
  ELSE
    0
  THEN
;
VARIABLE DELBYTE VARIABLE DELBIT
: DELMASK (    – – 2 ** DELBIT   )
  1
  DELBIT @ IF (   si DELBIT no es cero   )
    DELBIT @ 0 DO (   multiplicar 1 por 2   )
        2*          (   2 veces DELBIT   )
    LOOP
  THEN
;
: OTRO-MULTIPLO (    – – 0 ó –1   )
( regula DELBYTE y DELBIT )
( al siguiente bit si establecido )
( Queda 0 si se ha llegado al fin de la matriz BITS )
  PRIMOMOD8 @ DELBIT +!
  PRIMO/8 @ DELBYTE +!
  DELBIT @ 7 ) IF
      –8 DELBIT +!
      1 DELBYTE +!
  THEN
  DELBYTE @ BITSEND U< BITS
  1– DELBYTE @ U< AND
( atención en caso de que DELBYTE pase de
  65535)
;
: BORRAR-MULTIPLOS (    – –    )
( prepara bits para n/múltiplos de primo actual )
  CURBYTE @ DELBYTE !
  CURBIT @ DELBIT !
  BEGIN
  OTRO-MULTIPLO WHILE
    DELBYTE @ C@ DELMASK OR DELBYTE @ C!
    REPEAT
;
: PRIMOS (    – –    )
( imprime todos los primos entre 2 y 65535 )
  SETUP
  BEGIN
  OTRO-PRIMO WHILE
      PRIMO@ U.
      BORRAR-MULTIPLOS
  REPEAT
;
```

Palabras aritméticas

Suma y resta: no se hace distinción entre números con signo y sin signo. + y – son de longitud simple, D+ y D– de longitud doble.

El menos unario (un solo número negado): NEGATE para longitud simple, DNEGATE para doble longitud. El figFORTH utiliza MINUS y DMINUS.

Multiplicación: * es de longitud simple, con o sin signo. UM * es de longitud «mixta» sin signo: toma de la pila dos números sin signo de longitud simple y deja su producto en doble longitud. En los FORTH más antiguos UM* se denomina U*, y pueden tener un producto M*, de longitud mixta con signo.

División: /, MOD y /MOD son de longitud simple con signo. Dejan el cociente, el resto y ambos (el cociente arriba). UM/MOD es una versión de longitud mixta sin signo de /MOD. Su dividendo (el número que se divide) es de doble longitud. En los FORTH más antiguos se denomina U/ o U/MOD.

Entre el FORTH-83 y los FORTH más antiguos existe una diferencia en la forma de dividir números negativos. Pensando en hacer la división exactamente y después redondear a algún entero, el FORTH-83 redondea en un entero que no sea mayor; por ejemplo, 10/3 se redondea a 3, pero –10/3 se redondea a –4. Los FORTH más antiguos redondean hacia el 0, de modo que –10/3 se redondea a –3. Una vez calculado el cociente en función de estas reglas, el resto se determina mediante la fórmula:

dividendo = divisor*cociente+resto

Multiplicación combinada con división: */ es de longitud simple con signo, teniendo el siguiente efecto sobre la pila:

n1,n2,n3 – – n1* n2/n3

Comparaciones: = <> y U< son de longitud simple. < y > son con signo, U< es sin signo, = cualquiera. D= D< y DU < son de doble longitud, D< es con signo, DU< sin signo y D= cualquiera. MAX y MIN (que dan el mayor o menor de dos números) son de longitud simple con signo. DMAX y DMIN son de doble longitud con signo.

Valor absoluto: ABS y DABS son de longitud simple y doble.

Operaciones booleanas entre dos bits: AND, OR, XOR y NOT son todos de longitud simple. El signo no tiene importancia.

Combinaciones estándares: algunas palabras se utilizan juntas tan habitualmente, que sus combinaciones sin un espacio se definen como una única palabra. Por ejemplo, 1+ tiene exactamente el mismo efecto que 1+ (aunque se podría hacer con mayor eficiencia).

Impresión: . y U. son de longitud simple, con signo y sin signo, respectivamente. D. es de doble longitud con signo.

Direccionamiento de memoria: una dirección, de longitud simple sin signo, puede referirse a los dos bytes que haya en esa dirección (como con @ y !), a un solo byte de éstos (para lo cual se utiliza C@ y C!, donde la C es de «carácter») o al número de longitud doble de cuatro bytes que haya allí (use 2@ y 2!)

Quién, dónde y cómo

Ya estamos en condiciones de darles algún sentido a las vidas de nuestros personajes



David Higham

Comienza el juego
Vemos las posiciones iniciales de los distintos personajes según lo que establecen los valores por defecto en las líneas 6030 y 6040. También se aprecian los objetos, cuyos datos están retenidos en las líneas 6140 y 6150. Comienza el juego en la sala de tertulia

El primer módulo de nuestro programa comienza en la línea 10, inicializando la pantalla y DIMensionando las matrices que analizamos en el capítulo anterior. Luego lee los datos necesarios de las sentencias DATA correspondientes, todos los cuales están almacenados en el módulo DATA que empieza en la línea 6000. A estos dos módulos, por supuesto, se les irán añadiendo más líneas a medida que vayamos agregando otras rutinas e implementando el manipulador de personajes propiamente dicho. Una de las ventajas de adoptar este enfoque modular es que podremos ejecutar (RUN) y probar esta sección del programa apenas la hayamos digitado, y después ir probando cada una de las rutinas individualmente, tal como las vayamos entrando más adelante.

Obsérvese el aviso «Valores por defecto» de la línea 70. Éste, al ejecutar (RUN) el programa, nos permitirá entrar nuestros propios valores para los atributos de los distintos personajes en lugar de los valores por defecto entrados en las sentencias DATA. Esta facilidad de «confección a medida» es muy importante cuando se diseñan personajes interactivos con más de dos o tres atributos, porque permite «ir afinando» los personajes durante el juego. Por ejemplo, quizá encuentre que Mari Tapas es demasiado temperamental, o tal vez que pasa muy rápidamente de un escenario a otro.

La línea 530 es simplemente un bucle de comprobación que llamará, de a una, a cada una de las rutinas principales entradas hasta el momento. La rutina de entrada de la línea 2030 aguarda que se pulse una tecla. Se ignoran todas las teclas que no sean 0, 1, 2 o 3 (valores ASCII del 48 al 51) y se devuelve el control al bucle del programa de la línea 530. Pulsando 1, 2 o 3 el jugador penetra en el escenario especificado, donde podrá ver qué está sucediendo. Usted podría, por supuesto, diseñar una rutina de entrada que le permitiera dirigirse a los personajes directamente, pero en nuestro programa de ejemplo toda la interacción personaje/jugador la llevará a cabo el manipulador de personajes, debido a que hemos incluido al jugador como si se tratara de un personaje extra.

En caso de que usted decidiera adaptar los métodos que estamos analizando aquí para incluirlos en su propio juego de aventuras, recuerde que el manipulador de personajes puede existir independientemente de la rutina de entrada, como veremos más adelante cuando examinemos cómo se podría

Salón (escenario 2)
Sala de tertulia (escenario 1)
Cocina (escenario 3)

Kevin Jones

añadir una rutina manipuladora simple a nuestro programa *El bosque encantado*.

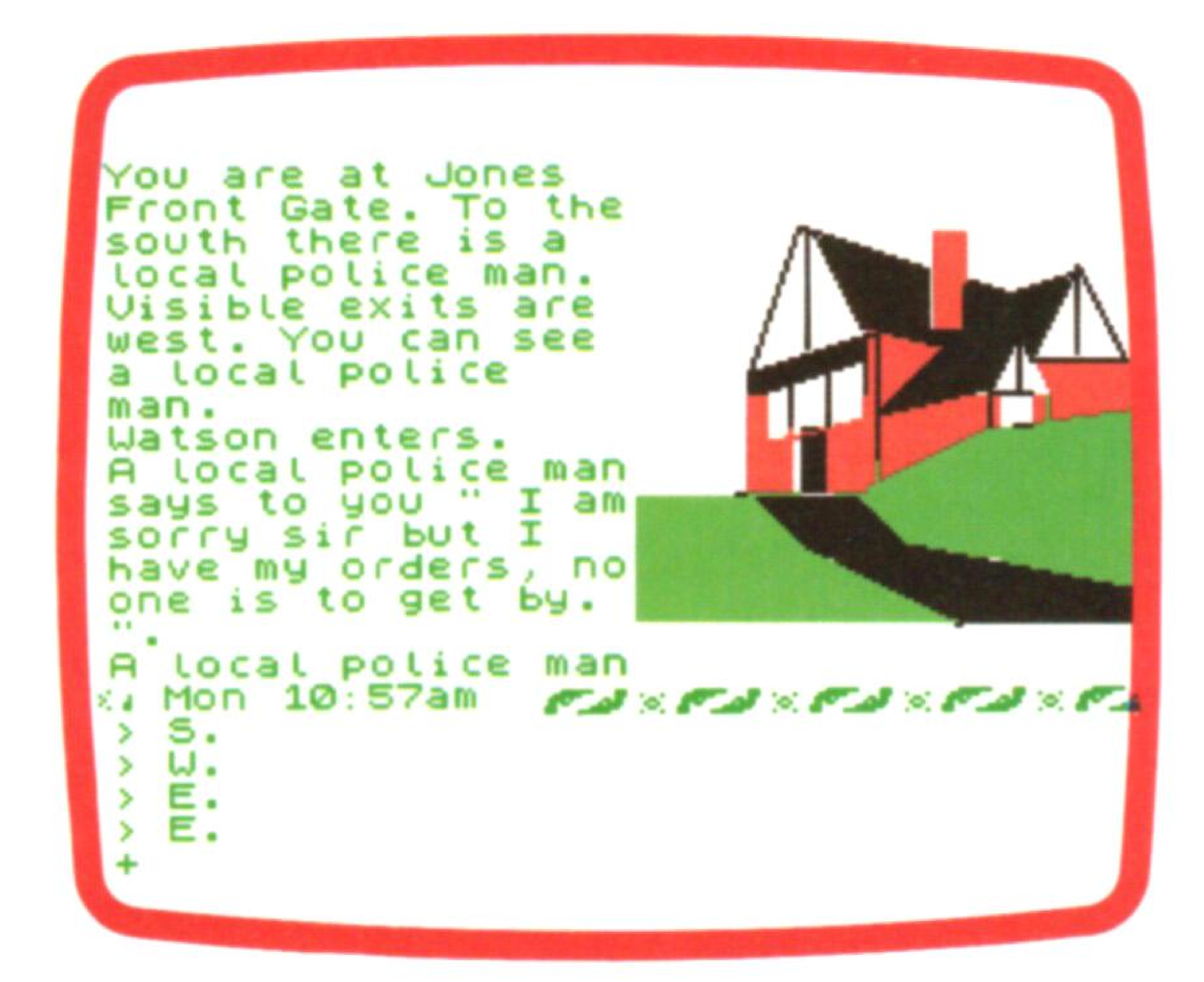
Pulsando la tecla 0 en cualquier momento, se entra en la rutina «Valores por defecto» de la línea 2300, que permite editar los atributos de los personajes como ya hemos mencionado y descrito.

Las líneas 2070-2390 proporcionan las tres subrutinas de visualización esenciales. La primera, de la línea 2100, imprime la descripción del escenario actual. Cuando ejecute el juego por primera vez, ésta será la descripción de la sala de tertulia, escenario número 1, como se establece en la línea 60 del módulo de inicialización.

La rutina de la línea 2150, «Imprimir objetos visibles», realiza un bucle a través de la matriz de objetos b$ y comprueba cada entrada de escenario para ver si corresponde con el escenario actual del jugador (r). Luego se imprimen en la pantalla las descripciones de objetos apropiadas; de lo contrario, se le dice al jugador «Ves: nada». La rutina «Imprimir personajes visibles» opera de forma similar, comprobando el elemento correspondiente de la matriz c$ e imprimiendo el nombre de un personaje si la persona en cuestión está presente.

La subrutina «Inicializar personajes» se llama cada vez que se pulsa 0 durante el tiempo de ejecución, o si durante la secuencia de inicialización se entra algo distinto a s o S en respuesta al aviso «Valores por defecto». Esta rutina utiliza la matriz d$(11) y el nombre de cada personaje para imprimir un aviso y pregunta si usted desea establecer los valores de la persona en cuestión. Si responde en sentido afirmativo, se visualizan los distintos atributos, cuyos títulos están almacenados en la matriz d$, y un valor entrado para cada uno. Al entrar una serie nula para un atributo (pulsando sólo Enter), saltará hasta

Detective con fallos
Sherlock, de Melbourne House, constituye un buen ejemplo de un programa que ofrece avanzadas facilidades interactivas totalmente en RAM, sin tener que cargar datos adicionales desde disco o cassette. El jugador puede charlar con los personajes utilizando «Decirle a...» y «Háblame acerca de...» y debe resolver varios delitos. Las primeras versiones del programa incluían algunos divertidos errores en la manipulación de personajes; en virtud de uno de estos «gazapos», el doctor Watson y Holmes estaban sentados en la misma silla simultáneamente



Módulo de inicialización

Este módulo irá experimentando adiciones a medida que se introduzcan nuevas variables y matrices

```
10 REM *bienvenido al Dog and Bucket*
20 REM
30 REM.........inicializar........................................
40 REM
45 GOSUB 4030:REM limpiar la pantalla
50 DIM 1$(3,5),b$(12,4),c$(7,11),d$(11)
60 r=1
70 PRINT «Valores por defecto (s/n)?":GOSUB 4110:IF i$="s"
   OR i$="S" GOTO 90
80 GOSUB 2300:GOTO 100
90 FOR n=1 TO 7:READ c$(n,1):FOR d=2 TO 11:READ
   c$(n,d):NEXT d:NEXT n
100 FOR n=1 TO 3:READ 1$(n,1):FOR e=2 TO 5:READ
   1$(n,e):NEXT e:NEXT n
110 FOR n=1 TO 12:READ b$(n,1):FOR d=2 TO 4:READ
   b$(n,d):NEXT d:NEXT n
120 FOR n=2 TO 11:READ d$(n):NEXT n
```

Bucle del programa

En futuros capítulos iremos alterando esta línea para que llame rutinas nuevas a media que las vayamos añadiendo

```
500 REM
510 REM bucle del/programa de prueba
520 REM
530 GOSUB 2100:GOSUB 2150:GOSUB 2240:GOSUB
   2030:PRINT: PRINT:GOTO 530
```

Módulo subrutinas de bajo nivel

Compruebe el recuadro *Complementos al* BASIC para su micro en particular; la versión impresa es para el Amstrad 464/664 y BASIC compatibles

```
2000 REM
2010 REM rutina de entrada
2020 REM
2030 GOSUB 4110
2040 IF (ASC(i$)<48) OR (ASC(i$)> 51) THEN
     RETURN
2050 IF i$=0 THEN GOSUB 2320:RETURN
2060 r=VAL(i$):c$(7,2)=i$:RETURN
2070 REM
2080 REM impresión escenario
2090 REM
2100 PRINT 1$(r,1)
2110 RETURN
2120 REM
2130 REM imprimir objetos visibles
2140 REM
2150 PRINT «Ves: »;
2160 p=0: FOR b=1 TO 12: IF VAL(b$(b,2))<>r GOTO 2190
2170 p=p+1: IF p> 1 THEN PRINT «, »;
2180 PRINT b$(b,1);
2190 NEXT b
2200 IF p=0 THEN PRINT «nada»;
2210 PRINT: RETURN
2220 REM
2230 REM imprimir personajes visibles
2240 REM
2250 p=0: FOR c=1 TO 6:IF VAL(c$(c,2))<>r GOTO 2290
2260 p=p+: IF p=1 THEN PRINT «Estás en compañía de:
     »;:GOTO 2280
2270 PRINT «,»;2280 PRINT c$(c,1);
2290 NEXT c
2300 IF p=0 THEN PRINT «Aquí no hay nadie.»
2310 RETURN
2320 REM
2330 REM subrutina inicializar personajes
2340 REM
2350 PRINT: RESTORE: FOR c=1 TO 6:READ c$(c,1):GOSUB
     4070:PRINT c$(c,1);« – »;:PRINT «Establecer este
     personaje?»: GOSUB 4110
2360 IF (i$<>«s») AND (i$><«S»)THEN FOR n=2 TO
     11:READ n$:NEXT n: GOTO 2390
2370 FOR d=2 TO 11:READ s$:PRINT d$(d);:INPUT i$:IF i$<>
     «» THEN c$ (c,d)=i$
2380 NEXT d
2390 NEXT c: RETURN
```

el siguiente, dejando sin modificar el valor original. Obsérvese, no obstante, que si se llama esta rutina durante la inicialización, se debe entrar un valor para cada atributo y se deben inicializar todos los personajes, ya que de lo contrario el manipulador de personajes no estará en condiciones de funcionar correctamente.

El módulo de «subrutinas de bajo nivel» que empieza en la línea 4000, se utiliza fundamentalmente para simplificar la conversión a máquinas diferentes.

Los datos se almacenan en último lugar. Naturalmente, a medida que programemos otras rutinas iremos introduciendo considerables cosas en este módulo. En particular, necesitaremos almacenar los distintos mensajes para los personajes.

Una vez que haya entrado el listado, ejecútelo (RUN) y experimente pasando de un escenario a otro pulsando las teclas 1, 2 y 3. También puede comprobar el funcionamiento del editor de personajes pulsando 0 y cambiando las entradas para los escenarios de los personajes, llevándolos de una habitación a otra y verificando que sus nuevas posiciones se visualicen correctamente.

Ahora ya disponemos de un «entorno» en el que dar cabida a nuestros personajes, y dentro del cual podemos verlos en funcionamiento. En el próximo capítulo veremos cómo vamos a «darles vida».

Módulo subrutinas principales

En primer lugar, sólo tenemos cuatro subrutinas principales: entrada, imprimir escenarios, personajes y objetos, y la subrutina «editora de atributos de los personajes»

```
4000 REM
4010 REM subrutinas de bajo nivel del sistema
4020 REM
4030 REM limpiar la pantalla
4040 REM
4050 CLS:RETURN
4060 REM
4070 REM bip
4080 REM
4090 PRINT CHR$(7);:RETURN
4100 REM
4110 REM tomar un carácter del teclado
4120 REM
4130 i$=INKEY$: IF i$=«» GOTO 4130
4140 RETURN
```

Módulo de datos

Observe que se han formateado las descripciones de los escenarios para una visualización a 40 columnas. Quizá deba modificarlas para adecuarlas a su micro. Por supuesto, podría alterar cualquiera de los valores por defecto de las líneas 6030-6040, en particular los valores entrados para «Ud.»

```
6000 REM
6010 REM datos de los personajes
6020 REM
6030 DATA "Luis Cubas", "2", "7", "10", "10", "7",
     "v", "0", "0", "7", "4", "Lola Fiestas", "1",
     "8", "30", "10", "8", "m", "0", "0", "3", "5",
     "Pepe Viñas", "1", "9", "8", "10", "9", "v",
     "0", "0", "4", "6", "Mari Tapas", "2", "0",
     "20", "10", "10", "m", "0", "0", "5", "5"
6040 DATA "Javi Salado", "2", "11", "10", "6", "11",
     "v", "0", "0", "4", "6", "Gina Fizz", "1", "12",
     "15", "6", "12", "m", "0", "0", "5", "5",
     "Ud.", "1", "0", "255", "255", "0", "v", "0",
     "0", "0", "0"
6050 REM
6060 REM datos escenarios
6070 REM
6080 DATA "Ud. se halla en la sala de tertulia del Dog
     and Bucket. En un rincón hay varios personajes
     dudosos jugando al dominó. Detrás del mostrador
     está Fred, el barman, con su habitual aspecto
     festivo. Las salidas dan al este.", "0", "0", "2", "0"
6090 DATA "He aquí el salón del Dog and Bucket, al cual
     le vendría muy bien una redecoración completa.
     Parece que el suelo ha sido regado regularmente
     con cerveza derramada. Hay puertas hacia el oeste
     y hacia el sur.", "0", "3", "0", "1"
6100 DATA "Uf! Ésta es la cocina, donde se preparan las
     famosas empanadas de carne del Dog and Bucket,
     para una clientela que siempre está famélica. Ud.
     observa numerosas latas vacías de alimento para
     gatos, lo cual no deja de ser extraño, ya que no hay
     ningún gato.", "2", "0", "0", "0"
6110 REM
6120 REM datos objetos (para b$(12,4))
6130 REM
6140 DATA "un vaso de cerveza", "2", "n", "s", " una
     lata vacía de alimento para gatos", "3", "n", "n",
     una empanada de carne del Dog and Bucket",
     "1", "s", "n", " una banqueta de bar",
     "2", "n", "n", "un cenicero", "1", "n", "n"
6150 DATA "un bocadillo de jamón rancio",
     "2", "s", "n", "una pinta de cerveza amarga",
     "0", "n", "s", "una crema de menta",
     "0", "n", "s", "un whisky con agua",
     "0", "n", "s", "un vodka puro",
     "2", "n", "s", "una pinta de cerveza añeja",
     "0", "n", "s", "una ginebra con ginger ale",
     "0", "n", "s"
6160 REM
6170 REM datos atributos personajes (para d$(11))
6180 REM
6190 DATA "Escenario", "Fortaleza", "Inventario",
     "Humor", "Objeto que tiene", "Sexo", "Ultimo
     personaje (lch)", "Código última instrucción
     (lcd)", "Frecuencia de manipulación", "Frecuencia
     de movimiento"
```

Complementos al BASIC

Spectrum:

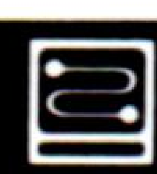
```
50   DIM L$(3,5,255), b$(12,3,25),
     c$(6,11,15),
     d$(11,25)
2040 IF(CODE(i$)< 48)OR (CODE(i$)> 51)
     THEN RETURN
4090 BEEP .5,3:RETURN
```

Commodore:

```
4050 PRINT"<shift/CLR> ":RETURN
4090 POKE 54296,15:POKE 54276,17:POKE
     54277,64
4095 FOR A=1TO50:POKE 54273,36:POKE
     54272,85:NEXT:POKE 54273, 0:POKE
     54272,0:POKE 54276,0:RETURN
4130 GET i$:IF i$="" THEN 4130
```

BBC:

```
4130 i$=GET$
```



Datos básicos (VII)

He aquí el análisis del siguiente fragmento del mapa de memoria del Commodore 64

ETIQUETA	DIRECCIÓN HEXA	POSICIÓN DECIMAL	DESCRIPCIÓN
MEMSTR	0281–0282	641–642	Puntero: parte inferior de memoria para el OS
MEMSIZ	0283–0284	643–644	Puntero: parte superior de memoria para el OS
TIMOUT	0285	645	Flag: variable núcleo para el Timeout IEEE
COLOR	0286	646	Código color carácter actual
GDCOL	0287	647	Color de fondo bajo cursor
HIBASE	0288	648	Parte superior memoria pantalla (página)
XMAX	0289	649	Tamaño buffer teclado
RPTFLG	028A	650	Flag: tecla REPEAT usada, $80=Repeat
KOUNT	028B	651	Contador velocidad repetición
DELAY	028C	652	Contador demora repetición
SHFLAG	028D	653	Flag: tecla SHIFT/tecla CTRL/ C=Tecla
LSTSHF	028E	654	Última configuración de shift
KEYLOG	028F–0290	655–656	Vector: establece tabla teclado
MODE	0291	657	Flag: $00=desactiva teclas SHIFT, $80=activa teclas SHIFT
AUTODN	0292	658	Flag: desplazamiento pantalla hacia abajo, 0=ON
M51CTR	0293	659	RS–232:imagen registro control 6551
M51CDR	0294	660	RS–232:imagen registro instrucciones 6551
M51AJB	0295–0296	661–662	RS-232: BPS no estándar (tiempo/ 2-100) en USA
RSSTAT	0297	663	RS232: imagen registro estado del 6551
BITNUM	0298	664	RS-232: número de bits que quedan por enviar

Acción industrial

Comenzamos una breve serie en la que esbozaremos el importante papel que desempeñan actualmente los ordenadores en la industria

Para ser eficaz, el proceso de manufacturación (desde el diseño del producto hasta la distribución de los productos acabados) exige conjugar hábilmente numerosos factores: tiempo frente a personas, máquinas frente a material, diseño frente a entrega y una docena de otras consideraciones. A principios de los ochenta, sólo los ordenadores más grandes eran suficientemente potentes como para ser de utilidad en las industrias. Gradualmente, esto ha ido cambiando, no sólo porque los pequeños ordenadores se han hecho más potentes, sino también porque las empresas han comenzado a tomar mayor conciencia de las ventajas que pueden aportar los ordenadores.

Es posible, aunque en muy pocos casos, ver un producto que ha sido realizado con el uso de ordenadores en todas las fases de fabricación, desde la etapa de diseño hasta el ensamblaje por robot. La mayoría de estas plantas avanzadas, como cabría suponer, pertenecen a la propia industria del ordenador. Las empresas de microelectrónica y los fabricantes de ordenadores personales figuran entre quienes marchan a la vanguardia de esta tecnología.

El empleo de ordenadores en todas las etapas del proceso de fabricación se conoce como CIM (*computer integrated manufacturing*: fabricación integrada por ordenador), pero la nomenclatura que se utiliza en esta área de aplicaciones puede ser muy desconcertante, como podemos ver en el recuadro *Jerga industrial*.

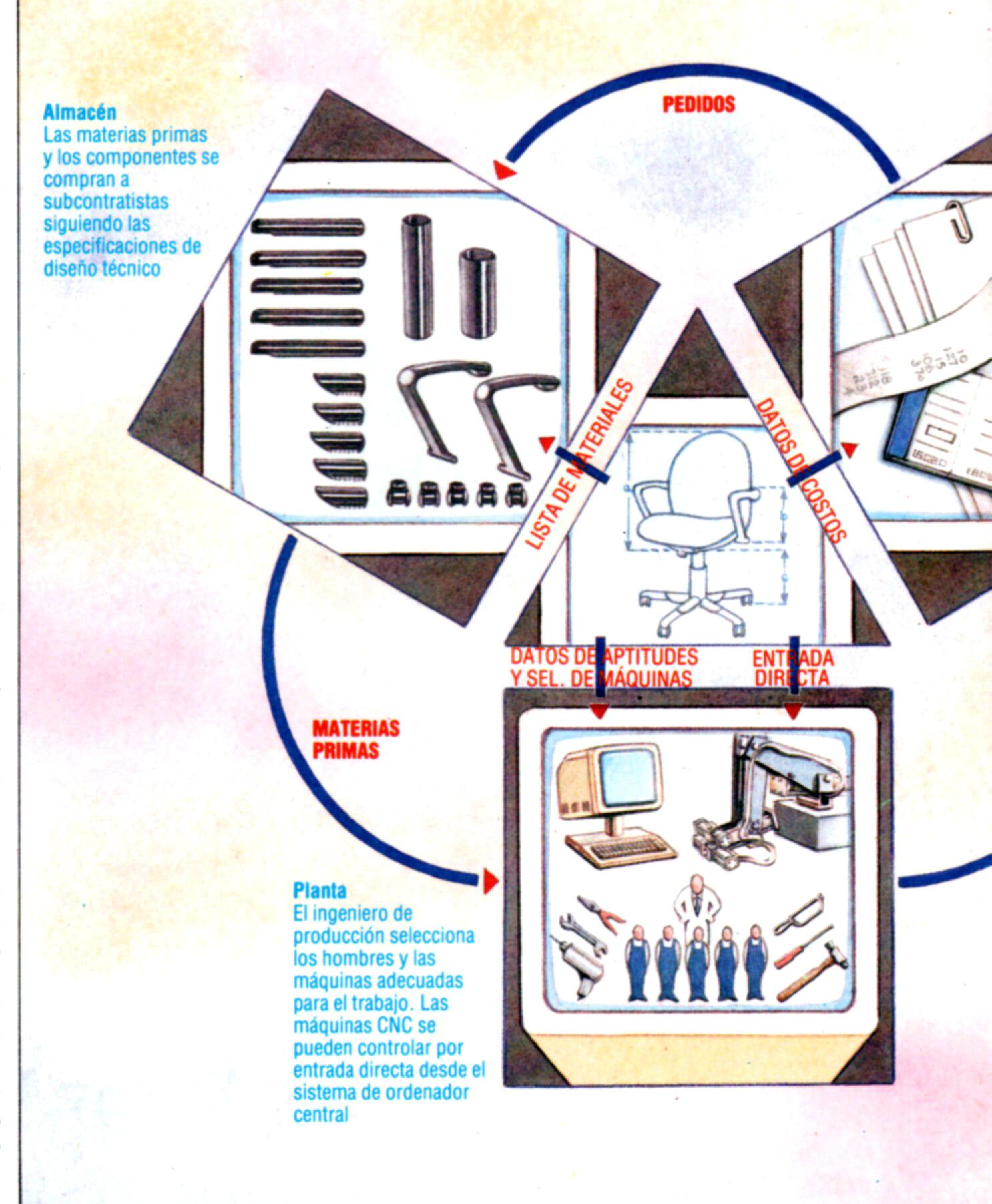
Los fabricantes trabajan sobre una base cooperativa, siendo cada departamento responsable de una parte separada del proceso. El departamento de diseño y dibujo mecánico desarrolla una idea y estipula las especificaciones técnicas que servirán de base a todo el proceso. Se habrán de pedir (y posiblemente fabricar) las herramientas para moldear el producto, y se habrán de programar las máquinas para producirlo. El almacén asegura que haya una constante disponibilidad de stocks de los materiales necesarios.

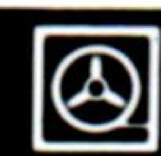
Luego está el aspecto financiero de la operación. Antes de pedir un producto, el consumidor desea saber el precio que tendrá y el fabricante ha de proporcionar una cotización exacta. Si ésta es demasiado elevada, el cliente irá a buscar a otro sitio, y si es demasiado baja, la empresa perderá dinero. Se debe aceptar el pedido, especificando una fecha de entrega y entregar el producto puntualmente, cobrándolo al final del proceso. El responsable de la planta ha de preocuparse por el despliegue de personal: cuántas personas se necesitan, qué aptitudes deben poseer y cuándo se necesitarán, así como asegurarse de que todos los pedidos aceptados se entreguen en las fechas especificadas.

Compartiendo datos

Por distintas que puedan parecer estas tareas, todas se basan esencialmente en los mismos datos. Ésta es la clave del enfoque CIM, en el que el ordenador retiene la base de datos que proporciona el núcleo de la información, y los distintos departamentos la utilizan en la medida de sus necesidades. El almacén puede consultar la descripción del producto y ver de qué está hecho, el encargado puede consultar qué máquinas se requieren para construirlo y los departamentos financieros pueden consultar cuán-

Valioso aporte
La informática puede aportar al proceso de manufacturación un grado de integración que hace un tiempo era impensable. El concepto CAD constituye la viga maestra de la organización de una fábrica informatizada, desde la etapa de diseño hasta el producto acabado





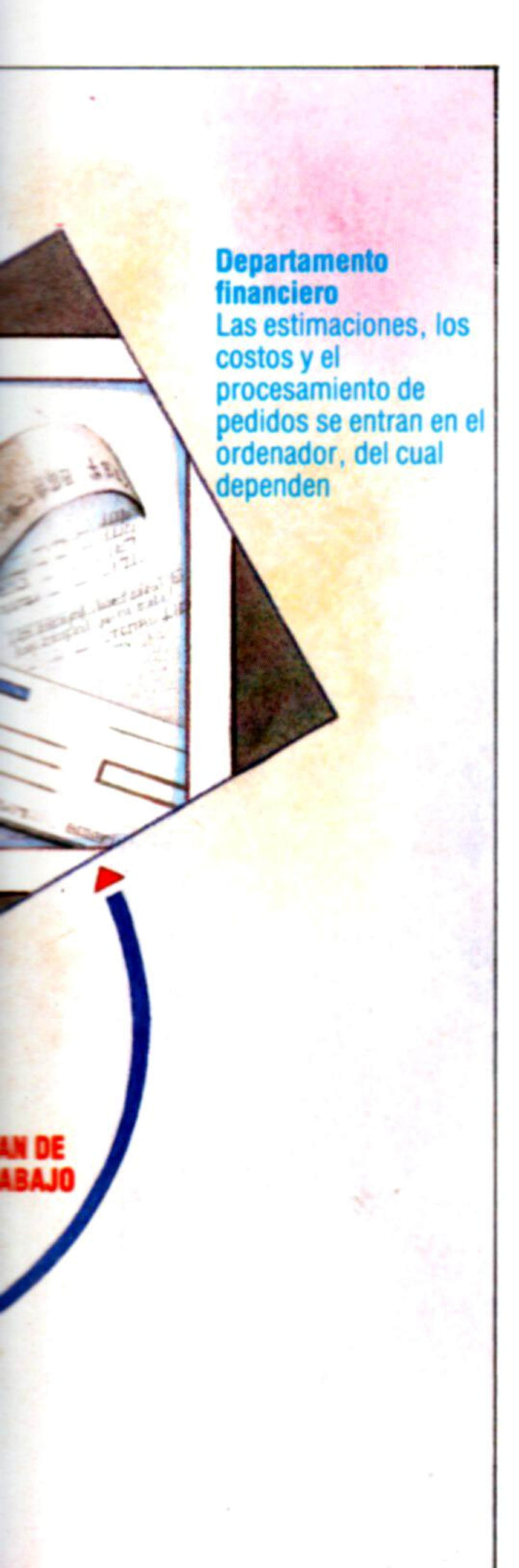
to cuesta producirlo: toda esta información se incluye en la base de datos.

Echemos ahora una mirada a los datos involucrados y los distintos pasos que supone su procesamiento. El dato crucial es, por supuesto, la *descripción del producto*, producida por el departamento de diseño. Ya hace muchos años que los sistemas CAD, de concepción asistida por ordenador, vienen ayudando en este proceso. Los productos compuestos a partir de un solo elemento son pocos; antes bien se componen de componentes hechos con componentes. Si tomamos un coche, por ejemplo, podemos ver que se trata de un diseño global, pero compuesto por millares de componentes distintos, desde el cableado de las luces traseras hasta el sistema de distribución del motor.

Los mejores sistemas CAD pueden formar una imagen del producto parte por parte, colocándolas en capas superpuestas hasta completar la imagen completa, en una representación tridimensional. A partir de esta imagen, el núcleo central de información, los programas de simulación y de análisis pueden comprobar no sólo el diseño, sino cómo funcionará el producto en la práctica.

La lista de materiales (*bill of materials*: BOM) es esencial para el correcto funcionamiento del almacén, al proporcionar una relación completa de todos los componentes del producto, que se puede descomponer en una lista de las materias primas necesarias. El interés del almacén consiste en cotejar la lista de componentes con la de los que hay en stock. El departamento querrá saber si todos los componentes están allí y si será preciso efectuar un nuevo pedido tras el inicio de la producción. De ser así, tendrán que informar al departamento de compras tanto de los suministros necesarios como de quiénes reciben habitualmente estas mercancías.

Los diseños del producto informan a los ingenieros de producción, responsables de la fabricación propiamente dicha, acerca de las necesidades de máquinas y de personal. A partir de este punto, está el pequeño paso de comparar las exigencias con el equipo y el personal disponibles y trazar un plan de trabajo.



Paul Bryant

La base de datos CIM

La CIM se utiliza no sólo para planificar la fabricación del producto, sino también para la planificación de varios productos diferentes que pasen todos por las diversas etapas de desarrollo de la misma fábrica. El sistema adecuado evitará situaciones en las cuales, por ejemplo, se hayan fabricado tornillos que sean inservibles por el mero hecho de que ninguna de las tuercas esté lista.

En otras palabras, entonces, la base de datos relativa a un producto, compilada en el momento del dibujo, puede detallar todos los procesos que entraña la construcción del mismo. La comparación de esta información con un plan de trabajo produce un programa laboral, y la misma base de datos informará a la oficina de contabilidad sobre los costos de producción. Medirá las cantidades de materiales a utilizar y, al planificar los tiempos-máquina y las aptitudes del personal, dará la información necesaria para proporcionarle al consumidor una cotización realista.

De los datos de la imagen precisa retenida en el dibujo, producido por el departamento de diseño, se puede pasar directamente a las máquinas controladas por ordenador que llevarán a cabo la operación. Ahora hay un tejido de información basado en las ideas originales de la oficina de diseño. El administrador del almacén, el departamento de herramientas, la oficina de ventas, los contables, el encargado e incluso la oficina de expediciones, todos ellos pueden sacar partido del mismo pozo de información relativa al producto.

En algunas de las industrias más nuevas los ordenadores han llegado a representar una parte tan activa del proceso de fabricación, que sin ellos no podrían continuar en funcionamiento. Por ejemplo, el diseño de los microprocesadores de hoy en día ya es demasiado complicado como para llevarlo a cabo sin ordenadores. El proceso llevaría demasiado tiempo si se realizara a mano, y sería imposible probar los diseños dibujados por los delineantes sin construir realmente el chip. Eso en sí mismo sería demasiado oneroso como para ser viable.

La CIM es la administración central de información sobre procesos de fabricación, y para ser eficaz requiere que se la alimente con información fiable. Con este fin se han desarrollado varias técnicas informáticas y en los próximos capítulos analizaremos algunas de ellas, así como las distintas formas en que pueden colaborar de manera insustituible en la industria.

Jerga industrial

Las aplicaciones de la industria basadas en ordenador están envueltas en el misterio de una jerga de siglas. He aquí algunos de los términos más utilizados:

AMT *Advanced Manufacturing Techniques*: técnicas avanzadas de fabricación. Engloba CAD, CAM y CAE

CAD *Computer Aided Design*: concepción asistida por ordenador. A veces se utiliza en alusión al dibujo técnico asistido por ordenador, una aplicación menos sofisticada capaz de dibujar imágenes pero no de analizarlas

CAM *Computer Aided Manufacture*: Fabricación asistida por ordenador. Se refiere a cualquier área en la que se utilicen ordenadores en el proceso de fabricación

CAE *Computer Aided Engineering*: ingeniería asistida por ordenador. El uso de datos (por lo general, la base de datos CAD) para llevar a cabo el análisis de un diseño

Máquina NC Máquina de control numérico. Utilizada para realización automática de piezas en el proceso de fabricación

Máquina CNC Máquina de control numérico por ordenador. Al igual que los ordenadores, las primeras máquinas NC solían ser activadas por instrucciones en cinta de papel. Ahora se pueden utilizar ordenadores para programar directamente las máquinas

CIM *Computer Integrated Management*: proceso de automatización de la administración de una planta industrial

BOM *Bill of Materials*: lista de materiales. Relación de componentes requeridos para la fabricación de un producto

Conciencia de grupo

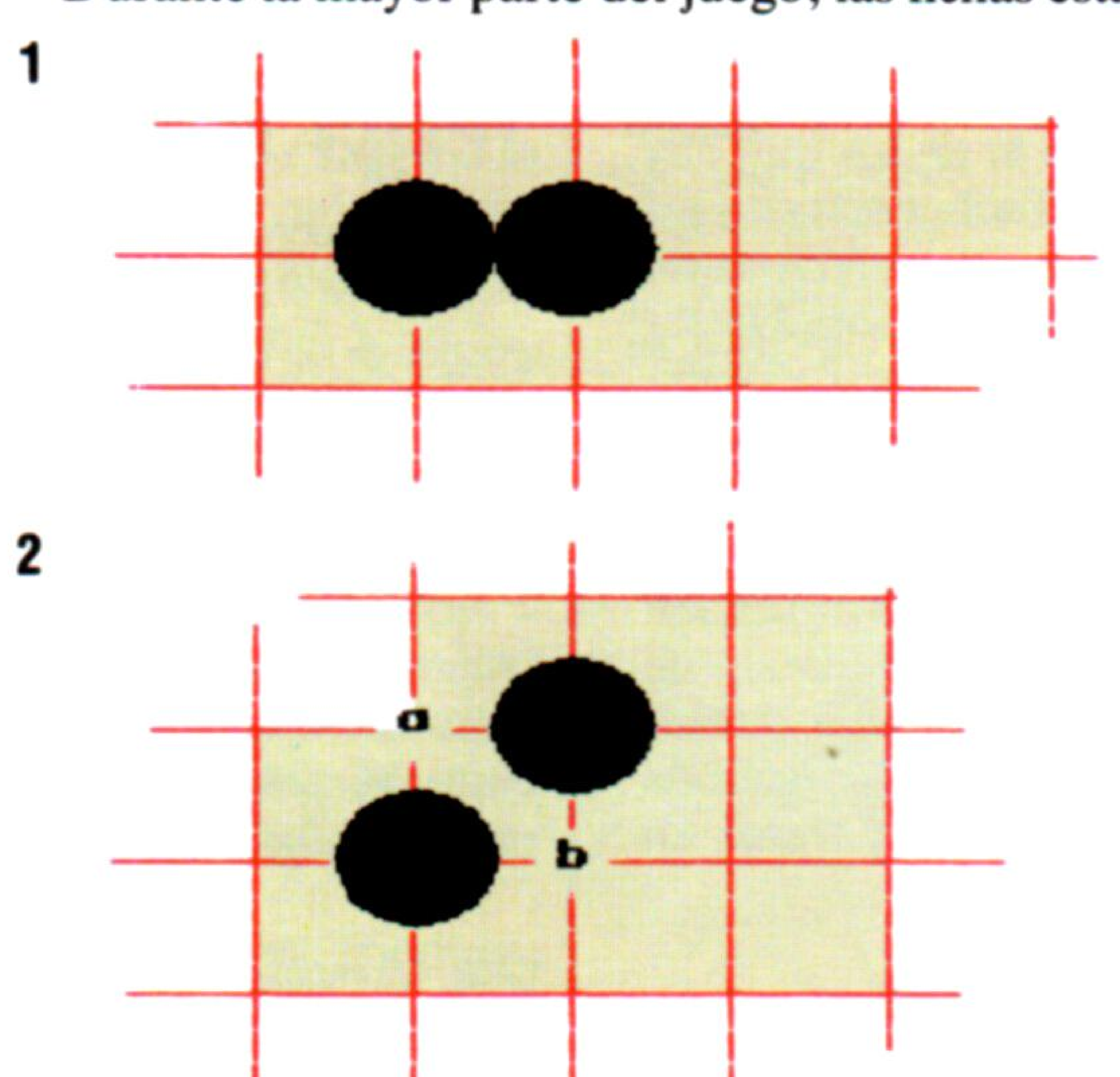
A partir de ahora nuestro programa tratará todas las posibles agrupaciones de fichas

Al terminar el juego, el territorio sólo está rodeado oficialmente si se halla delimitado por una línea continua de fichas. Ésta consistirá en uniones sin interrupción (fig.1) o en uniones en diagonal (fig. 2), donde si las blancas juegan a *a*, las negras pueden formar dos uniones sin interrupción jugando *b*, y viceversa. Sin embargo, por lo general estas uniones sólidas sólo se pueden formar al final de la partida.

Durante la mayor parte del juego, las fichas esta-

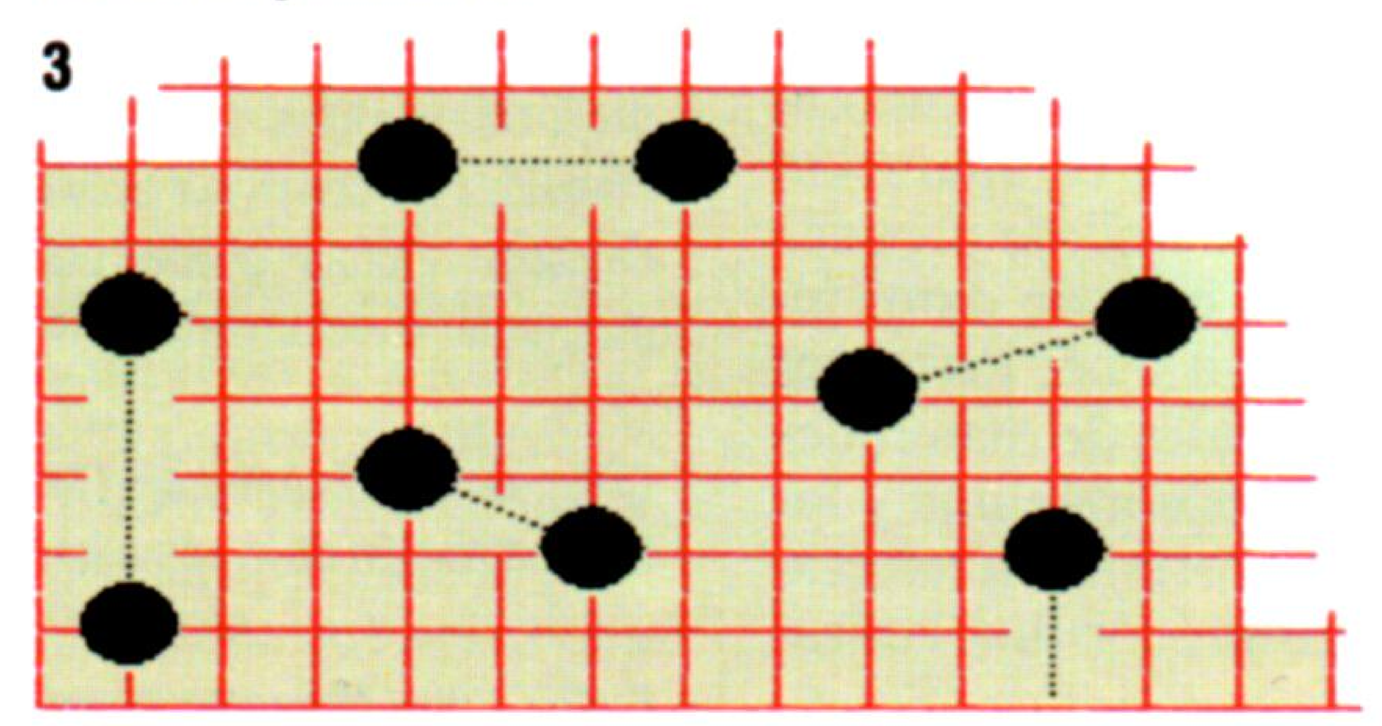
1

2



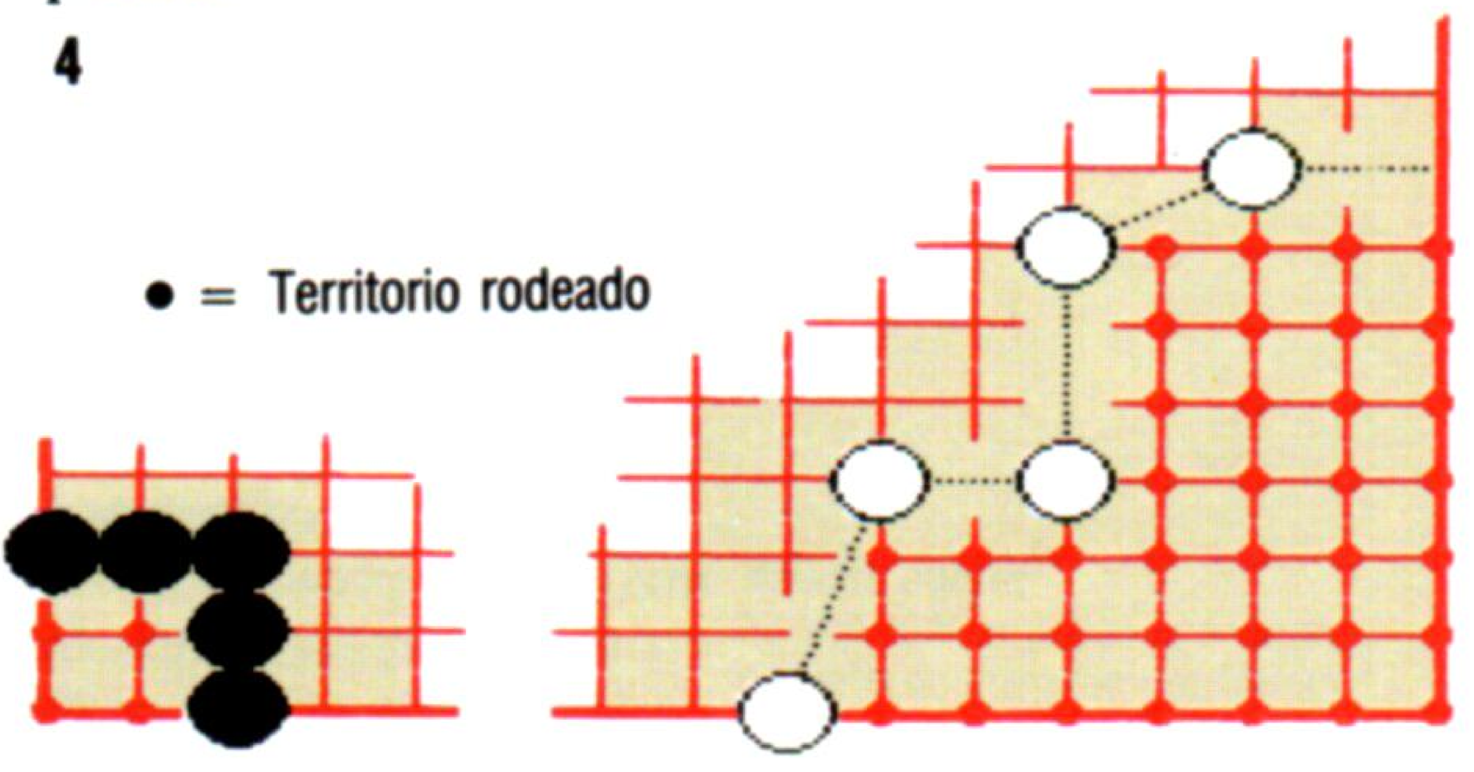
rán separadas por uno o más espacios, formando uniones imaginarias, bien con otras fichas, bien con el borde del tablero (fig. 3). En estos casos, sólo se formarán uniones sólidas cuando no exista ninguna posibilidad de que el oponente abra una brecha en estas «conexiones». No existe una garantía absoluta de que finalmente estas fichas se conecten, pero es bastante probable.

3



Obviamente, si queremos que nuestro programa juegue una partida aceptable, hemos de decirle algo acerca de estas posibles conexiones. Si sólo le enseñamos que para rodear territorio se pueden utilizar conexiones sólidas de fichas adyacentes, el programa se encontrará con problemas. Por ejemplo, en la figura 4, mientras que nuestro programa (negras) está rodeando cuatro puntos de territorio ¡el oponente ha empleado la misma cantidad de fichas para rodear un prometedor territorio de 37 puntos!

4

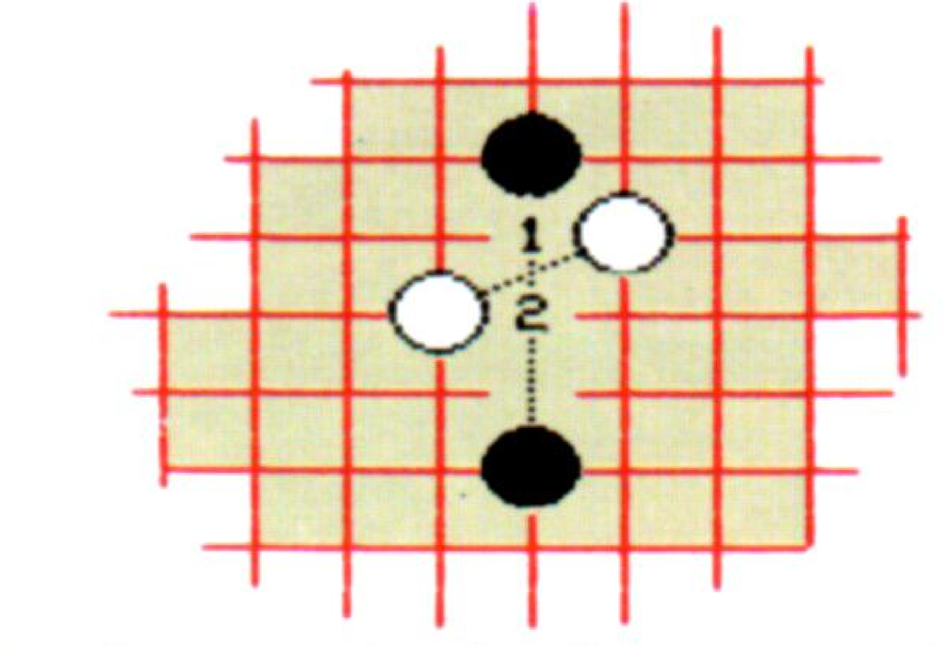


Habiendo decidido que el ordenador comprenda las conexiones, hemos de decidir cómo programarlo. Deseamos que manipule cuatro operaciones principales de conexión. Éstas son:

- Defender una conexión entre dos grupos de fichas cuando el oponente se mueva hacia una posible posición de ataque.
- Atacar una conexión del oponente colocando una ficha entre las fichas conectadas débilmente.
- Jugar una ficha de inicio de ataque. Ésta será una acción previa al ataque, colocando una ficha central, o bien proporcionará un apoyo para una ficha colocada en el centro.
- Iniciar una conexión colocando una ficha a cierta distancia de otra ficha colocada anteriormente. Mediante el empleo del sistema de estimación (desarrollado en el capítulo anterior) se evaluará esta ficha, y automáticamente se hallará una posición atinada para poder rodear territorio.

Estas operaciones se han programado utilizando técnicas sencillas de emparejamiento de formatos. Observe una situación típica como la que muestra la figura 5. Idealmente, las blancas desean romper

5



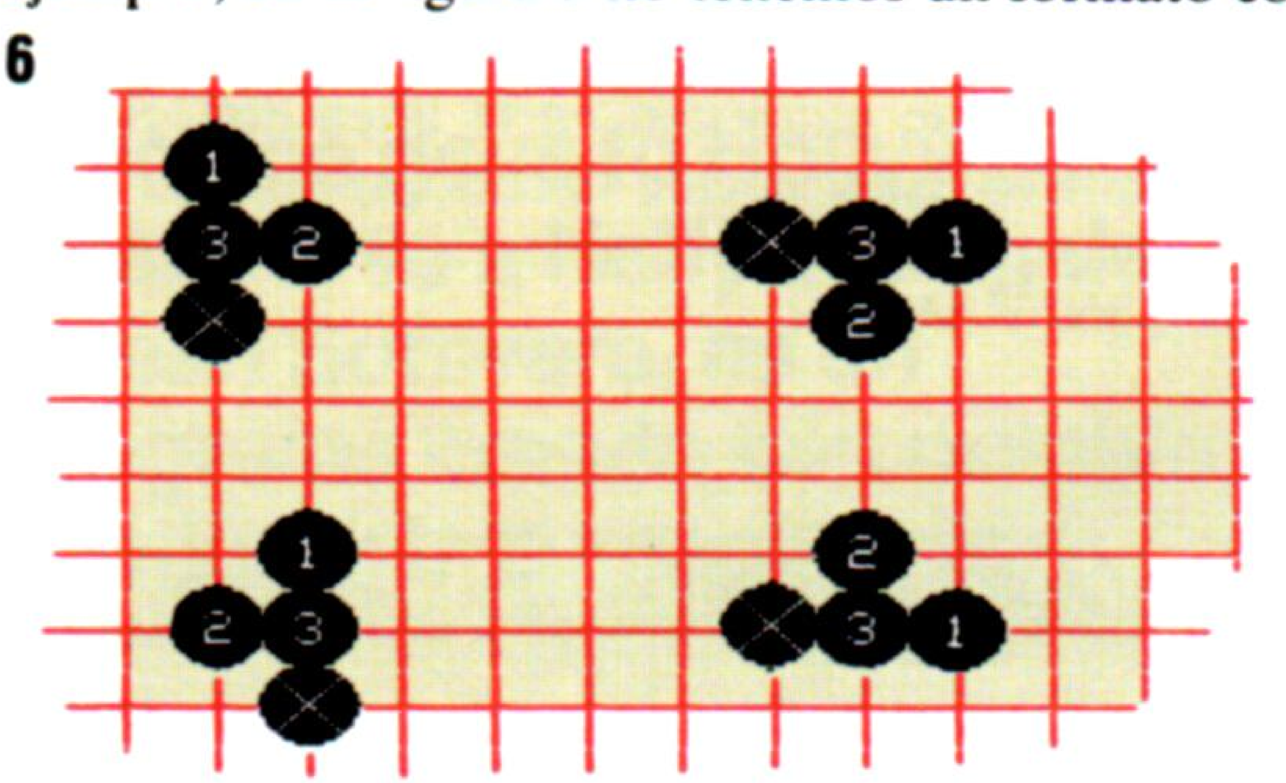
la conexión entre las dos fichas negras jugando en la posición «2», y las negras quieren defenderse jugando en la misma posición. Al mismo tiempo, las negras desean romper la conexión de la ficha blanca bien en «1», bien en «2», mientras que las blancas quieren defenderse jugando en estos puntos. En este caso particular, las blancas tienen ventaja al tener las fichas conectadas más estrechamente.

Es posible implementar los cuatro tipos de operaciones de conexión utilizando un conjunto único de formatos. Estos formatos se establecen mediante la rutina que comienza en la línea 790, que define los formatos ilustrados en las figuras 6, 7, 8 y 9. En todos los casos, la ficha con una cruz es nuestra

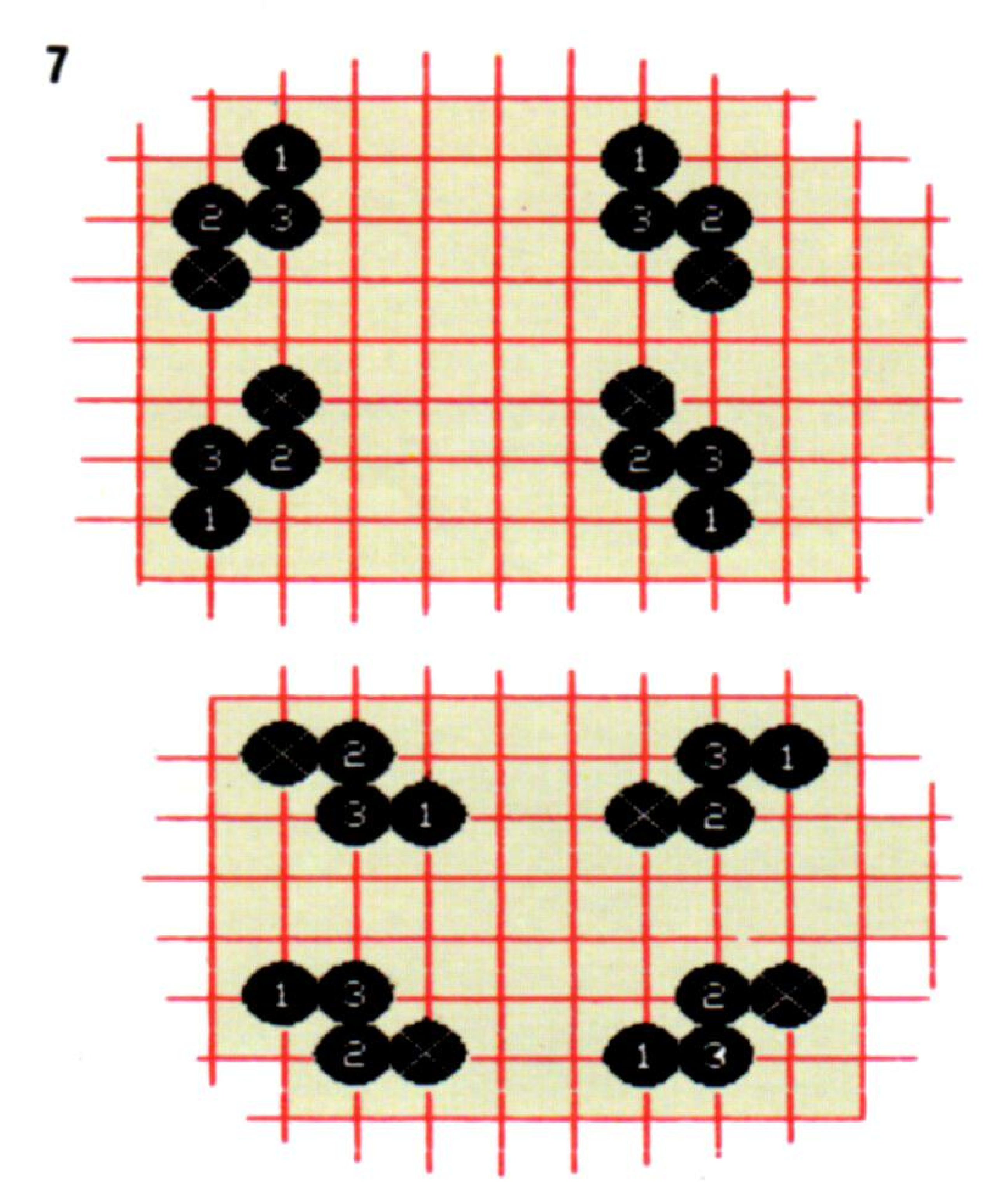
ficha actual (punto cero). Luego las sentencias DATA dan ramales desde este punto en el mapa del tablero%. De este modo, +16, −16, +1 y −1 corresponden respectivamente a norte, sur, este y oeste.

Puesto que a la larga todas las fichas se considerarán como la ficha actual, no es necesario implementar todas las orientaciones de cada patrón. Por ejemplo, en la figura 9 no tenemos un formato con

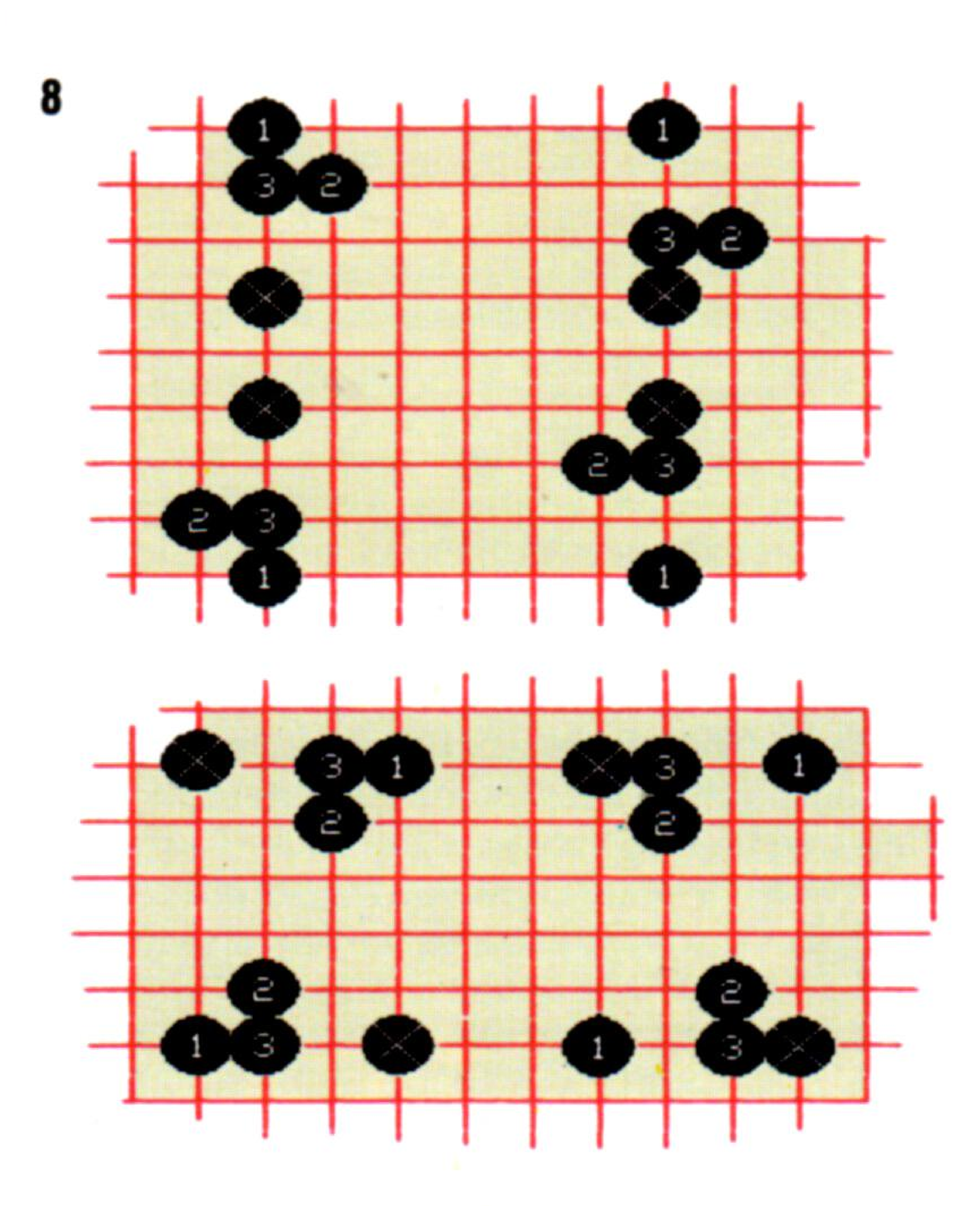
6



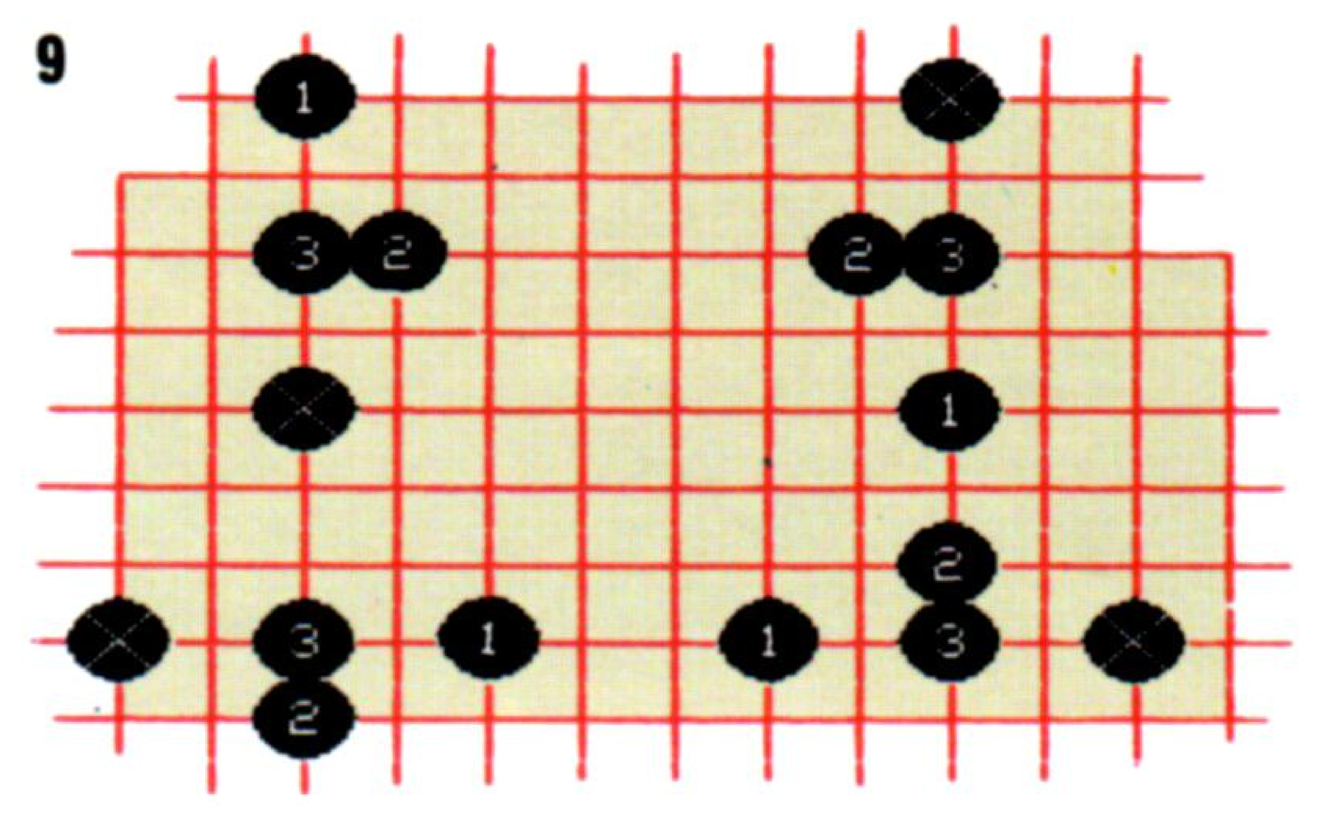
7



8



9



la ficha actual en la parte inferior (0), la primera ficha de la parte superior (+ 64) y la ficha 2 a la izquierda de la ficha 3 (+ 31 y + 32). No obstante, más adelante durante la investigación del tablero, la ficha que ahora está marcada como 1 se considerará como la ficha actual. El formato que vemos arriba y a la derecha de la figura 9 manipulará luego la orientación del formato que acabamos de describir. Por tanto, los formatos simétricos sólo requieren cuatro orientaciones en la memoria, y los formatos no simétricos requieren ocho.

Observará que todas las demás sentencias DATA no son más que la negación de la línea anterior, de modo que podríamos reducir aún más la cantidad de formatos si estuviéramos escasos de memoria, probando específicamente la negación de todos los formatos retenidos.

Las rutinas de emparejamiento de formatos trabajan calculando las posiciones de estos ramales en el tablero y, por tanto, hallando el color de las fichas, si las hubiera, en las correspondientes posiciones. Luego se aplica un conjunto de reglas para decidir si jugar o no una ficha. Si se va a jugar una ficha, a la posición se le asigna un marcador basado en las estimaciones del tablero; al final de la rutina, se utiliza la ficha que tenga el marcador más alto.

Las rutinas se comprueban por orden, de modo que si, por ejemplo, la rutina conexión_de_ataque encuentra un movimiento, entonces no se comprobarán las rutinas iniciar_ataque e iniciar_conexión.

Suponiendo que hubieran de jugar las negras, las reglas aplicadas para decidir si jugar o no una ficha son:

```
conexión_de_defensa:
   IF      ficha con cruz   =NEGRAS
   AND     ficha uno        =NEGRAS
   AND     ficha dos        =BLANCAS
   THEN    intentar movimiento de ficha tres.
conexión_de_ataque:
   IF      ficha con cruz   =BLANCAS
   AND     ficha uno        =BLANCAS
   AND     ficha dos        =NEGRAS
   THEN    intentar movimiento de ficha tres.
iniciar_ataque:
   IF      ficha con cruz   =BLANCAS
   AND     ficha uno        =BLANCAS
   AND     ficha tres       =LIBRE
   THEN    intentar movimiento de ficha dos.
iniciar_conexión:
   IF      ficha con cruz   =NEGRAS
   AND     ficha tres       =LIBRE
   THEN    intentar movimiento de ficha uno.
```

Las cuatro rutinas de conexión que vemos aquí se llaman desde las líneas 2580-2610 de la rutina movimiento_negras. Con ello ahora manipularemos las conexiones entre las fichas del tablero. No obstante, si usted vuelve a observar las figuras 3 y 4, verá que también tenemos conexiones entre fichas y el borde del tablero, que son igualmente importantes. Para tratar con ellas hemos diseñado la rutina PROCfrontera. Recordará que nuestro tablero realmente se define DIM tablero% 256 para un tablero de 15 por 15. Esto da una ocupación de 256 bytes, y también nos deja con un borde de una sola ficha alrededor de todo el tablero. PROCfrontera rellena este borde con el valor pasado como parámetro a V%. Al asegurar que las rutinas de emparejamiento de formatos busquen por todo el tablero, incluyendo el borde (de 1 a 255), también se comprobarán las conexiones con el borde del tablero. Acuérdese de restablecer siempre a cero el borde cuando salga de la rutina, llamando PROCfrontera con un valor de cero.

Debido a la velocidad del programa la cantidad de formatos se ha mantenido en un mínimo. Sin embargo, usted está en libertad de añadir datos o cambiar los que empiezan en la línea 860. Si por algún motivo usted decide hacerlo, no olvide cambiar los parámetros del bucle en las líneas 2940, 3110, 3260 y 3410.

Módulo 5

BBC Micro:

```
280 PROCleer_formatos
780 :
790 DEF PROCleer_formatos
800 LOCAL L%
810 DIM pat% 71
820 RESTORE 860
830 FOR L%=0 TO 71
840    READ for%?L%
850    NEXT
860 DATA 32,17,16,2,-15,1
870 DATA -32,-17,-16,-2,15,-1
880 DATA 33,16,17,31,16,15
890 DATA -33,-16,-17,-31,-16,-15
900 DATA -14,1,-15,18,1,17
910 DATA 14,-1,15,-18,-1,-17
920 DATA 48,33,32,48,17,16
930 DATA -48,-33,-32,-48,-17,-16
940 DATA 3,-14,2,3,-15,1
950 DATA -3,14,-2,-3,15,-1
960 DATA 64,33,32,4,-14,2
970 DATA -64,-33,-32,-4,14,-2
980 ENDPROC
990 :
1000 REM ******************************************
2580 IF posicion%=0 THEN PROCconexion_de_
     defensa:T$="DEF"
2590 IF posicion%=0 THEN PROCconexion_de_
     ataque:T$="ATT"
2600 IF posicion%=0 THEN PROCiniciar_ataque:T$="SAT"
2610 IF posicion%=0 THEN PROCiniciar_conexion:T$="SCN"
2890 :
2900 DEF PROCconexion_de_defensa
2910 LOCAL A%,B%,C%,D%,P%,S%,hi,marcador
2920 PROCfrontera(negras%)
2930 FOR A%=1 TO 255: S%=tablero%?A% : IF S% <>
     negras% THEN 3010
2940    FOR P%=for% TO for%+70 STEP 3
2950       B%=tablero%?((A%+?P%) AND
           255):C%=tablero%?((A%+?P%+1)) AND 255)
2960       IF B %<>negras% OR C%<>blancas% THEN
           3000
2970       D%=A%+?(P%+2)) AND
           255:marcador=RND(1)+estimaciones%?D%
2980       IF (D% AND 240)=0 OR (D% AND 15)=0 OR
           marcador< = hi THEN 3000
2990       IF FNlegalidad(D%,negras%)=0 AND clib%>2 THEN
           hi=marcador: posicion%=D%
3000       NEXT
3010    NEXT
3020 PROCfrontera(0)
3030 ENDPROC
3040 :
3050 REM ******************************************
3060 :
3070 DEF PROCconexion_de_ataque
3080 LOCAL A%,B%,C%,D%,P%,S%,hi,marcador
3090 PROCfrontera(blancas%)
3100 FOR A%=1 TO 255 : S%=tablero%?A% : IF S% <>
     blancas% THEN 3180
3110    FOR P%=for% TO par%+70 STEP 3
3120       B%=tablero%?((A%+?P%) AND
           255)):C%=tablero%?((A%+?(P%+1)) AND 255)
3130       IF B%<>blancas% OR C%<>negras% THEN 3170
3140       D%=(A%+?(P%+2)) AND
           255:marcador=RAN(1)+estimaciones%?D%
3150       IF (D% AND 240)=0 OR (D% AND 15)=0 OR marcador
           < = hi THEN 3170
3160       IF FNlegalidad(D%,negras%)=0 AND clib%> 2 THEN
           hi=marcador: posicion%=D%
3170       NEXT
3180    NEXT:PROCfrontera(0)
3190 ENDPROC
3200 :
3210 REM ******************************************
3220 :
3230 DEF PROCiniciar_ataque
3240 LOCAL A%,B%,C%,D%,P%,S%,hi,marcador
3250 FOR A%=1 TO 255 : S%=tablero%?A% : IF S% <>
     blancas% THEN 3330
3260    FOR P%=for% TO pat%+70 step 3
3270       B%=tablero%?((A%+?P%) AND
     255):D%=tablero%?((A%+?(P%+2)) AND 255)
3280       IF B%<>blancas% OR D%<> 0 THEN 3320
3290       C%=(A%+?(P%+1)) AND
           255:marcador=RND(1)+estimaciones%?C%
3300       IF (C% AND 240)=0 OR (C% AND 15)=0 OR
           marcador< = hi THEN 3320
3310       IF FNlegalidad(C%,negras%)=0 AND clib%> 2 THEN
           hi=marcador: posicion%=C%
3320       NEXT
3330    NEXT
3340 ENDPROC
3350 :
3360 REM ******************************************
3370 :
3380 DEF PROCiniciar_conexion
3390 LOCAL A%,B%,C%,P%,S%,hi,marcador
3400 FOR A%=1 TO 255 : S%=tablero%?A% : IF S% <>
     negras% THEN 3470
3410    FOR P%=for% TO pat%+70 STEP 3
3420       C%=tablero%?((A%+?(P%+2)) AND 255):IF C%>0
           THEN 3460
3430       B%=(A%+?P%) AND
           255:marcador=RND(1)+estimaciones%?B%
3440       IF (B% AND 240)=0 OR (B% AND 15)=0 OR marcador
           < = hi THEN 3460
3450       IF FNlegalidad(B%,negras%)=0 AND clib%>2 THEN
           hi=marcador: posicion%=B%
3460       NEXT
3470    NEXT
3480 ENDPROC
3490 :
3500 REM ******************************************
4410 :
4420 DEF PROCfrontera(V%)
4430 LOCAL X%,Y%
4440 FOR X%=0 TO 15
4450    Y%=16*X%
4460    tablero%?X%=V%
4470    tablero%?Y%=V%
4480    NEXT
4490 ENDPROC
4500 :
4510 REM ******************************************
4520 REM ********* FIN DEL PROGRAMA ***************
```

Video mágico

«Frankie goes to Hollywood» (Frankie va a Hollywood), de Ocean Software, es un programa creado con notable inteligencia e imaginación

La búsqueda de Frankie
Frankie goes to Hollywood tiene como objetivo buscar objetos en numerosas habitaciones. Cuando se abre un cajón, su contenido se visualiza en una ventana de la pantalla. También se puede «entrar» en ciertos cuadros y jugar a un juego de tipo recreativo

Liz Heaney

Frankie goes to Hollywood no se basa tanto en la música del grupo británico de igual nombre, sino más bien en su filosofía de escapar del mundo cotidiano («Mundanesville») hacia la cumbre del placer («Pleasure Dome»), y, en el camino, convertirse en una «verdadera persona».

El juego es un producto de Denton Designs, un equipo de programación de Liverpool, ciudad de la que también es originario el grupo. Contratado originalmente por Imagine, que cesó sus actividades en 1984, ahora el equipo trabaja en base a colaboraciones independientes para numerosas empresas de software, produciendo, entre otros programas *Shadowfire*, tan bien considerado.

Al tener numerosos puntos de semejanza con su antecesor, *Frankie...* es un juego de aventuras con numerosos lugares para explorar (en este caso una fila de casas con terraza). Mientras usted se va desplazando por las casas, puede ir revisando los muebles en busca de objetos que puedan serle útiles. Estos objetos están escondidos en cajones y otros lugares tan inverosímiles como las lavadoras.

La mayoría de las aventuras exigen que se digiten instrucciones, pero Denton Designs ha adoptado el método de explorar objetos que utilizara por primera vez en *Shadowfire*. Una vez que se ha hallado un objeto, se lo puede recoger o abandonar colocando el cursor sobre el mismo y pulsando el botón de disparo. Entre los objetos más comunes que se descubrirán hay cuatro pequeños iconos que corresponden a los cuatro componentes de su personalidad. Éstos se indican en el lado derecho de la pantalla y van aumentando a medida que el usuario se convierte en una «verdadera persona». Los iconos que corresponden a estos componentes son «píldoras de placer» que, al tomarlas, aumentan su porcentaje de ese aspecto de su personalidad.

El jugador sólo puede llevar consigo ocho objetos al mismo tiempo y, a medida que su inventario va creciendo, tiene la tentación de tomarse las píldoras apenas las encuentra. Sin embargo, éstas son más útiles en una etapa ulterior del juego y, por ello, es interesante conservarlas durante un tiempo.

Otro conjunto de objetos que aparece continuamente son videos que le permiten a usted entrar en una faceta del juego completamente distinta. La mayoría de las casas tienen aparatos de televisión y en ellos se pueden insertar las cassettes de video. Aparte de ganar numerosos puntos para el marcador, el video aparecerá en una ventana de la pantalla. Al desplazar su «persona» dentro de la pantalla de video, el programa cambia a uno de numerosos juegos de tipo recreativo. Es durante estos juegos cuando el comer las píldoras del placer hará que su marcador se dispare al máximo. En algunas de las casas hallará cuadros colgados en las paredes y también podrá entrar en ellos y jugar a un juego recreativo adicional.

En las casas se pueden encontrar, asimismo, varios discos flexibles. En un primer momento parecería que éstos no tuvieran ningún uso práctico, ya que tampoco parece haber ningún ordenador donde colocarlos. Sin embargo, escondida en algún lugar del juego hay una sala de ordenadores y en sus manos está hallarla. Como es natural, cuando finalmente descubre esta habitación, la información retenida en el disco le ayudará en su búsqueda para convertirse en una verdadera persona.

En el curso de sus viajes, usted descubre que se ha cometido un asesinato, y una de sus tareas consiste en identificar al criminal. A intervalos aparecen de forma intermitente en la pantalla pistas para resolver el asesinato. Por ejemplo, una pista podría ser El asesino es socialista, mientras que otra podría ser Miss Mundane siempre vota a los conservadores, con lo que ella quedaría eliminada de la lista de sospechosos.

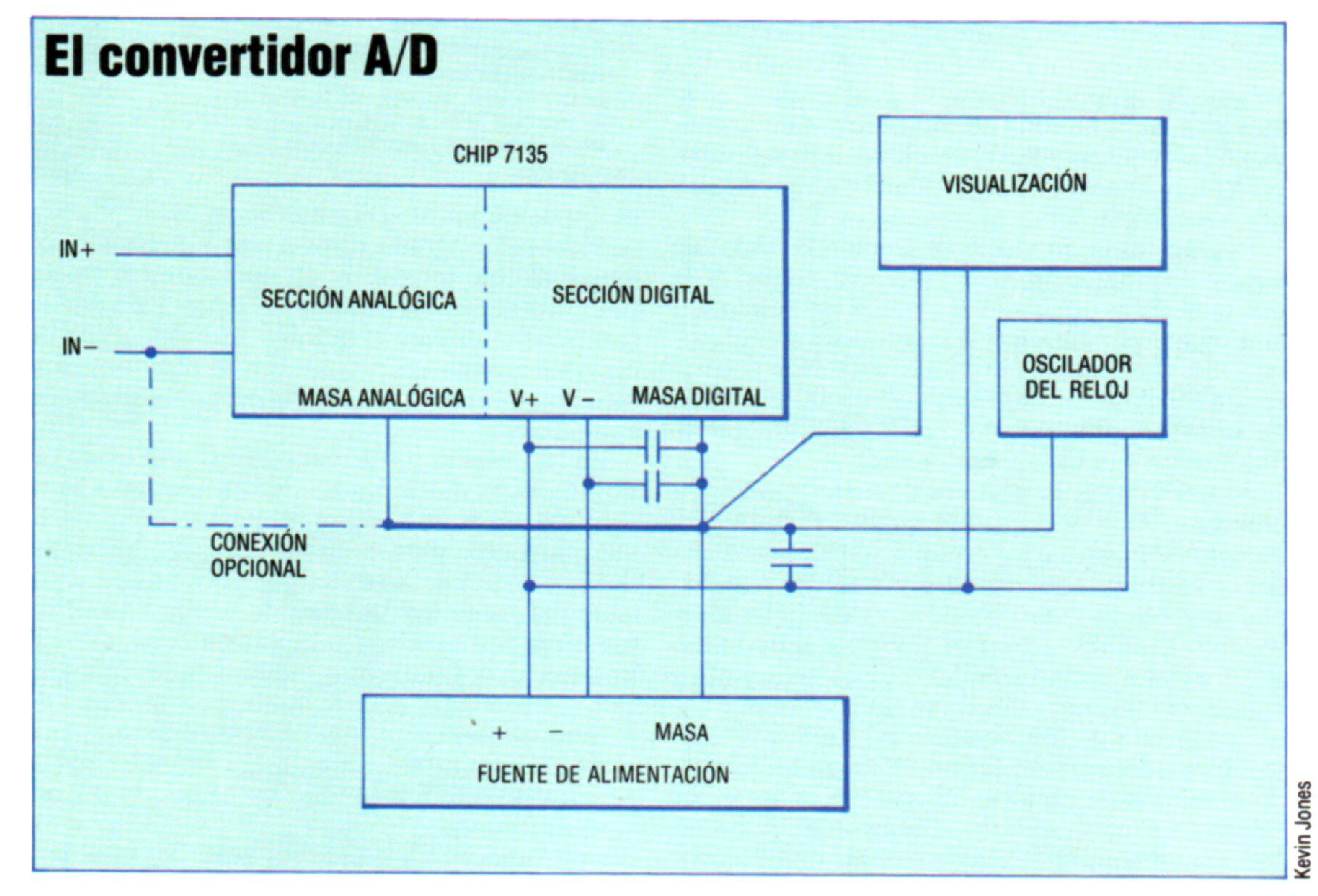
A pesar de que muchas de las facilidades de *Frankie goes to Hollywood* se han tomado prestadas de otros juegos, es decididamente bueno por derecho propio. La adición en un mismo programa de tantas facetas diferentes es ciertamente insólita, y entretendrá a los amantes de los juegos durante un tiempo considerable.

Frankie goes to Hollywood: Para el Sinclair Spectrum y el Commodore 64
Editado por: Ocean Software, Ocean House, 6 Central Street, Manchester, Gran Bretaña
Autores: Denton Designs
Formato: Cassette o disco
Palancas de mando: Necesarias

Centro vital

Examinemos el chip convertidor 7135 y estudiemos algunos puntos importantes referentes al trazado de la placa y la selección de componentes

Principios de la alimentación
Al considerar el trazado del circuito del DVM es muy importante asegurar que las secciones analógica y digital del diseño no entren en contacto entre sí. El esquema general de cableado que vemos aquí minimiza los bucles de masa y el acoplamiento accidental de señales digitales con el circuito analógico



Remitiéndonos, en primer lugar, al esquema del circuito realizado anteriormente, que muestra la disposición de las patillas del 7135 y sus respectivos nombres, iremos refiriéndonos a ellas por el orden de los números de patillas. Ulteriormente ofreceremos una lista completa de componentes.

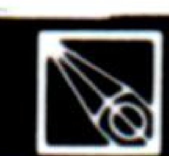
1. *V−*: el 7135 requiere una fuente de alimentación tanto positiva como negativa. Las *especificaciones máximas absolutas* de estas entradas son +6 V para la línea positiva y −9 V para la negativa. Para simplificar las exigencias de la fuente de alimentación, utilizaremos +/− 5V. La corriente de alimentación de −5 V típicamente es de 0,8 mA, de modo que todo cuanto se requiere es una fuente de alimentación muy pequeña y sencilla. En nuestro diseño, la unidad se alimentará con la red eléctrica, pero también se podría utilizar una pila de 6 V (tanto los chips MOS como los CMOS funcionan bien con una fuente de 6 V). En la fase de construcción de esta serie ofreceremos detalles relativos a la fuente de alimentación con la red eléctrica.

2. *Referencia*: La tensión de referencia para el 7135 (o para cualquier convertidor A/D de atenuación doble) es crítica para su precisión general. La desviación de fondo de escala (d.f.e.) —el máximo valor de salida— de un convertidor A/D de atenuación doble es el doble que la tensión de referencia, de modo que para un tester con una d.f.e. de 2 V, el voltaje de referencia ha de ser de 1,000 V.

Existen muchísimas formas de obtener una referencia exacta de 1 V, pero nosotros aconsejamos utilizar un diodo de referencia de intervalo de banda de precisión de 1,22 V, del tipo 9491. La referencia de 1 V requerida se obtiene luego utilizando un potenciómetro para reducir la tensión al nivel requerido. No caiga en la tentación de utilizar un potenciómetro barato del tipo de control de volumen. Utilice un potenciómetro de ajuste multivueltas (de diez o más vueltas) de la mejor calidad. Pida prestado un buen DVM y regule el potenciómetro de ajuste para 1,000 V. Mejor aún, monte un divisor de potencial en la salida de la fuente de alimentación de prueba de baja tensión, y regúlelo para obtener en el DVM prestado una lectura de 1,999 V; alimente éste a la entrada de su DVM y regule el potenciómetro de ajuste de referencia para una lectura de 1,999 V en su visualización.

Si no tiene acceso a un DVM de precisión, lo mejor que puede hacer es crear una fuente de tensión de precisión moderada utilizando dos células de mercurio de 1,35 V en serie y un divisor de potencial construido con una resistencia de 36 K-ohmios y una de 100 K-ohmios. La misma dará aproximadamente 2 V para utilizar como entrada de prueba para el DVM. De nuevo, regule el potenciómetro de ajuste de referencia para obtener en la visualización una lectura de 1,999 V. Emplee resistencias de estrecha tolerancia (1 % o mejores) y células de mercurio nuevas.

3. *Masa analógica*: La patilla del 7135, de apariencia inocua, es la que probablemente cause más

problemas que ninguna otra en su DVM. El problema proviene del hecho de que hay dos señales de masa separadas: masa analógica y masa digital (la masa digital está en la patilla 24). Las señales analógicas, alimentaciones + y −, y circuitos de masa de las secciones analógica y digital, deben mantenerse estrictamente separadas. Se requiere un sistema de cableado que minimice la posibilidad de que las señales digitales se introduzcan en la sección analógica y viceversa, si bien este último caso no reviste tanta importancia.

Los diseñadores de circuitos le dirán que debe adoptar buenos principios de trazado, pero esto no es de gran ayuda si usted no sabe cuáles son los principios de un buen trazado. No profundizaremos demasiado y nos limitaremos simplemente a decirle qué debe hacer y qué no debe hacer.

Todos los puntos de masa analógica (la unión del diodo de referencia y el extremo inferior del potenciómetro de ajuste conectado a la patilla 3) y el extremo a masa de la resistencia conectada al diodo (que está conectado a los condensadores de integración y de autocero en las patillas 4 y 5) se deben soldar en un mismo punto. Suelde el ánodo del diodo de referencia y la resistencia de 100 K-ohmios (conectada a *Int.Out.* a través de un diodo) a un punto único.

Medidas de precaución

Es igualmente importante conectar todos los puntos de masa digital en un mismo punto. Observe que el esquema del circuito utiliza un símbolo diferente para los dos tipos de masa: ⏚ para masas analógicas y ⏊ para masas digitales. Habiendo unido todas las conexiones de masa analógica en un único punto, y todas las digitales en otro punto único y distinto, se deben conectar entre sí las dos tierras. Esto debe hacerse utilizando un solo trozo de conductor de cobre macizo de gran sección.

De no respetarse estas precauciones se produciría una multitud de complicaciones. Las lecturas de la visualización serían inexactas, la visualización podría pasar por varias lecturas (erróneas) alternativas, el DVM podría no captar las lecturas, y todo oscilaría o se clavaría.

La fuente de alimentación para el reloj, el activador de la visualización y la propia visualización deben desacoplarse cuidadosamente empleando tanto condensadores electrolíticos de valor elevado como condensadores de disco cerámicos de bajo valor. Es importante que los componentes analógicos del circuito (como el diodo de la tensión de referencia, las resistencias integradoras, los condensadores, etc.) queden situados en la placa en el lado opuesto de los componentes digitales (tales como la señal CLOCK y las líneas STROBE y BUSY).

De no hacerse así, es probable que señales como BUSY y STROBE, que en el curso de una lectura pasan de un nivel lógico a otro, se acoplen con la entrada y provoquen errores. La regla es: mantener los componentes analógicos y digitales del circuito separados físicamente en la placa, y prestar atención al trazado de las líneas de masa de modo que los bucles no creen problemas adicionales. El diagrama *Principios de la alimentación* ilustra la forma general en que debe plantearse la alimentación.

En tercer lugar, no se debe dejar flotando ninguna señal digital inutilizada (tales como STROBE, UNDER-RANGE, RUN/HOLD, etc.); se las debe dejar altas empleando resistencias de 4,7 K-ohmios o bien dejar bajas mediante resistencias de 440 ohmios.

4. *Int.Out.(Salida integradora)*: Los convertidores A/D de tipo integrador como el 7135 se basan en el supuesto de que el cambio de tensión en el condensador integrador es exactamente proporcional a la corriente que circula en él. El valor absoluto de este condensador no es muy crítico (0,4 o 0,47 μF [microfaradios], p. ej.), pero es vital que tenga unas pérdidas dieléctricas muy bajas. Los condensadores de polipropileno son los mejores, pero los de poliestireno son casi igual de buenos, y también se pueden emplear los de policarbonato. No utilice ninguna clase de condensador electrolítico.

5. *A-Z IN*: El condensador utilizado aquí puede ser de cualquier calidad. El valor no es crítico, si bien cuanto mayor sea, tanto mejor. Emplee condensadores de polipropileno, poliestireno o policarbonato, pero no electrolíticos.

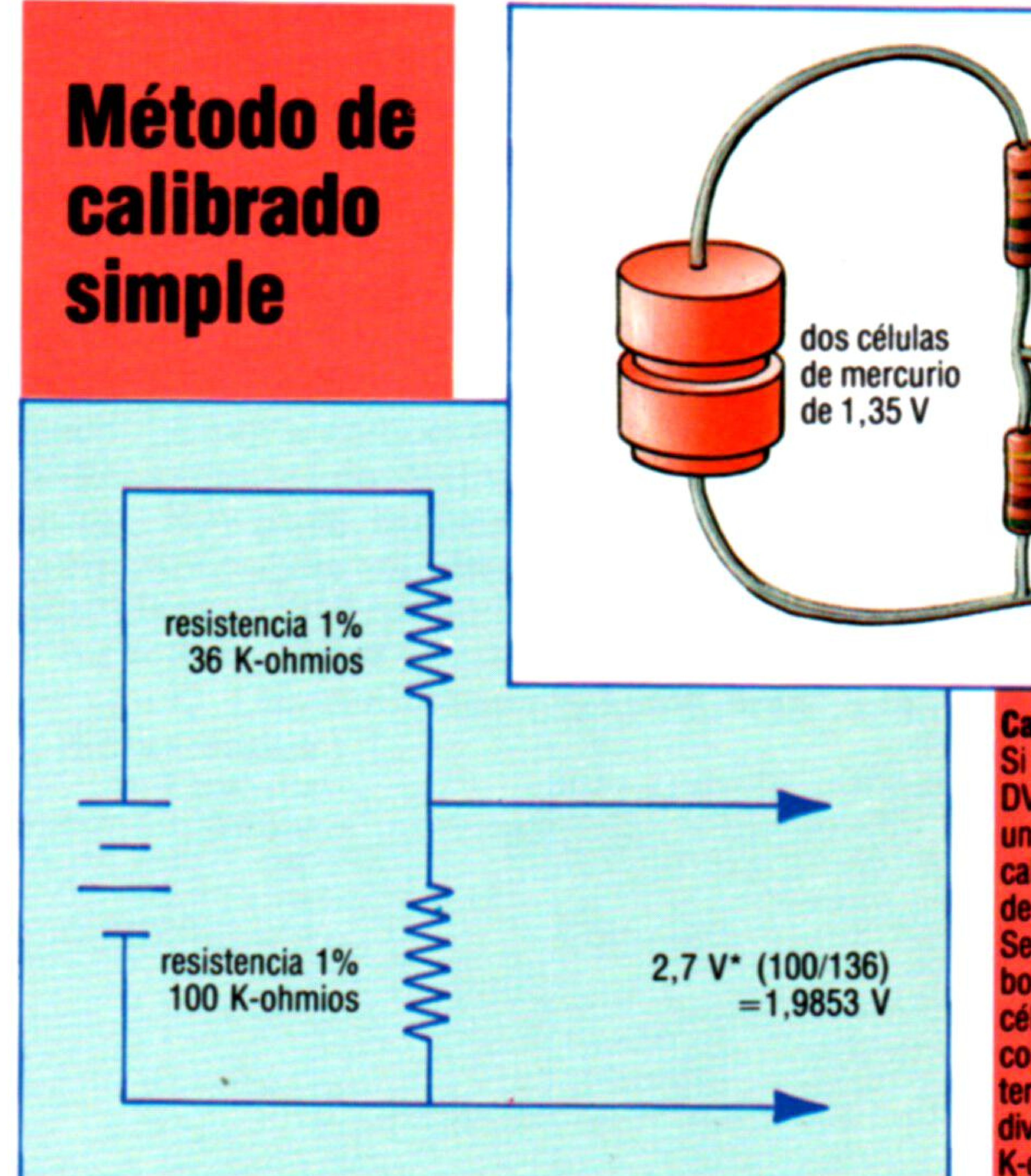
6. *BUF OUT (Buffer Output)*: Éste es el punto donde se conecta la resistencia integradora. El valor de ésta se obtiene con:

$$R = \frac{\text{tensión de fondo de escala}}{20\ \mu A}$$

Servirá cualquier resistencia de 100 K-ohmios, si bien una resistencia de película metálica de estrecha tolerancia (1% o mejor) sería mejor que las resistencias de carbón de baja tolerancia que uno suele tener olvidadas en el fondo del cajón.

El diodo conectado a la salida integradora (patilla 4) es para corregir un pequeño error de deriva. Servirá uno de silicio de pequeña señal de cualquier tipo, de los que suele haber montones en los cajones. Aunque por lo general a los diseñadores de circuitos poco experimentados les agrada que se les recomiende un tipo específico, casi cualquier diodo servirá, aunque ha de ser de silicio y no de germanio, y de pequeña señal en vez de rectificador.

Método de calibrado simple



Calibrado simple
Si durante la construcción de su DVM usted no puede acceder a un DVM de precisión, podrá calibrarlo utilizando esta fuente de tensión de precisión media. Se requieren dos células de botón de mercurio (no utilice células de óxido de plata). Se conectan en serie para dar una tensión de 2,7 V mediante un divisor de potencial de 136 K-ohmios. Teóricamente, la tensión en la resistencia de 100 K-ohmios será de 1,9853 V, pero podría oscilar de forma imprevisible alrededor de +/− un 2% de esta cifra, siempre que se utilicen resistencias del 1% de tolerancia. Existen a la venta resistencias económicas de película metálica del 1% dentro de estos valores. 36 K-ohmios es uno de los valores de resistencias de la «gama ampliada». No intente construirlo con resistencias de la «gama estándar», o correrá el riesgo de perder precisión. El potenciómetro de ajuste deberá regularse hasta que la visualización dé una lectura de 1,9999 V

Nitidez

Nuestro estudio del sistema operativo del Spectrum concluye con un examen de la pantalla

El manejo de los gráficos y de la pantalla no es tan difícil en el Spectrum. En primer lugar examinaremos cómo se usan las rutinas que están en la ROM y que son empleadas por el sistema operativo.

Para direccionar la pantalla primero debe abrirse el canal S seleccionando el flujo 2. Para una operación de impresión (PRINT) normal, el método más fácil es, sin duda, la rutina RST#10. Pero si deseamos imprimir una cadena entera de caracteres, ha de escribirse una pequeña rutina para entrar cada carácter uno a uno, y pasarlos también uno a uno a la rutina RST#10 hasta agotar toda la impresión.

No obstante, en lugar de escribir una rutina propia, es preferible usar la que ya existe en la ROM encargada de hacer esa tarea. Para llamarla, basta con cargar DE con la dirección en la que hay almacenada la cadena, cargar BC con la longitud de la cadena y llamar la rutina Print String que está en #203C. Véase una muestra de lo dicho en el siguiente listado:

```
                ORG  60000
1172EA          LD   DE,MSG
3E02            LD   A,2          ; selecciona flujo 2
CD0116          CALL #1601        ; abre canal S
1172EA          LD   DE,MSG       ; apunta DE a la cadena
012600          LD   BC,38        ; carga BC con la long. ca-
                                    dena
CD3C20          CALL #203C        ; llama PRINT-STRING
C9              RET
160A06    MSG:  DEFB 22,10,6      ; códigos para AT 10,5
54484520        DEFM «VAN TRES»
160C07          DEFB 22,12,7
41445641        DEFM «DEL APOCALIPSIS»
```

Existe otra rutina ROM muy útil: la Print a Floating Point Number (impresión de un número en coma flotante: abreviadamente, PRINT-FP), que está en #2DE3. Ya hemos mostrado cómo se evalúa una expresión numérica y cómo se coloca en la pila de cálculo empleando la rutina que está en #1C82. La rutina PRINT-FP nos puede facilitar las cosas, toda vez que imprimir un número en código máquina es una tarea bastante difícil. Para imprimir un número, primero hay que colocarlo en la pila de cálculo cargándolo, bien en el par de registros BC, bien en el registro A. La rutina STACK-A se llama en #2D28 y la rutina STACK-BC en #2D2B. Una vez en la pila, la impresión se logra con una sencilla llamada a la rutina PRINT-FP.

```
                ORG  60000
013930          LD   BC,12345     ; toma el número en BC
CD2B2D          CALL #2D2B        ; BC empujado a pila
                                    cálculo
3E02            LD   A,2
CD0116          CALL #1601        ; abre canal S
CDE32D          CALL #2DE3        ; imprime No. de encima
                                    pila
C9              RET
```

La verdadera potencia de los gráficos del Spectrum se hace más patente cuando empleamos las instrucciones de alta resolución PLOT y DRAW. Ya tratamos de la primera. Pues bien, DRAW es tan fácil como PLOT, salvo que son necesarios unos cuantos parámetros más. Ya sabemos que la instrucción tiende a trazar una línea lo más directa posible entre el último punto trazado y un punto definido según el origen. Así, si hubiere que dibujar una línea 20 pixels hacia arriba y 40 pixels a la izquierda, la instrucción sería DRAW −40, 20.

A fin de explicar el empleo de la instrucción DRAW en código máquina, nos serviremos de la expresión sintáctica DRAW x, y. En la ejecución de DRAW en código máquina, los parámetros x e y deben considerarse por separado. Para simplificar el tema, de momento nos despreocuparemos de cualquier factor para curvas. Si ha de hacerse un retorno al BASIC en un determinado punto, habrá que conservar el valor del par alternativo de registros H'L', dado que la rutina DRAW alterará los datos en él contenidos.

El valor absoluto de x (ABS x) se lleva al registro C, mientras que ABS y se pasa al registro B y los valores de signos de y y de x se llevan a la rutina en D y E respectivamente. Pueden emplearse valores superiores a −1, 0 y 1, puesto que la rutina emplea estos valores como incremento efectivo que se añade al último pixel trazado para alcanzar la posición del siguiente pixel de la línea.

Esto significa que si se utilizase un valor de signo 2 o 3, la línea sería dos o tres veces más larga de lo normal, pero se trazaría con un aspecto similar al que se puede obtener en el QL. Obsérvese que los valores de signos que emplea la rutina han de escribirse en complemento a dos.

En el tercer listado pueden verse ejemplos de ambos tipos de dibujo.

```
          ORG  60000
D9        EXX
E5        PUSH HL           ; guarda H'L'
D9        EXX
0658      LD   B,88         ;coordenada Y
0E64      LD   C,100        ;coordenada X
CDE522    CALL #22E5        ;PLOT 100,08
0E28      LD   C,40         ;ABS x=40
0614      LD   B,20         ;ABS y=20
1EFF      LD   E,255        ;SGN y=−1
1601      LD   D,1          ;SGN x=1
CDBA24    CALL #24BA        ;DRAW −40,20
3E58      LD   A,88
327D5C    LD   (23677),A    ;X org=88
3E32      LD   A,50
327E5C    LD   (23678),A    ;Y org=50
0E28      LD   C,40
0614      LD   B,20
1602      LD   D,2          ;SGN x=2 línea de puntos
1E02      LD   E,2          ;SGN y=2 línea de puntos
CDBA24    CALL #24BA
D9        EXX
E1        POP  HL           ;restaura H'L'
D9        EXX
C9        RET
```

La información que ofrecemos en dos cuadros sintetiza todo lo que hemos estudiado en esta parte sobre gráficos y manejo de la pantalla. El primer cuadro contiene una lista de las rutinas más útiles que hemos empleado, y el segundo muestra las variables de sistema más comúnmente relacionadas con esas rutinas.

El archivo de visualización

La memoria de pantalla del Spectrum se organiza en dos partes. La primera llamada *archivo de visualización (display file)* se emplea para retener toda la información sobre los pixels actuales. La segunda parte, el *archivo de atributos* (*attributes file*), almacena la información sobre el color en la pantalla. El archivo de visualización va desde 16384 hasta 22527, y comprende un total de 6144 bytes. Escriba este programa y ejecútelo:

```
10 FOR F=16384 TO 22527
20 POKE F,255
30 NEXT F
```

A medida que el programa se va ejecutando, verá cómo la pantalla se llena lentamente. Mire más detenidamente y verá que el relleno se divide en tres zonas. Y apurando la observación de la pantalla podrá observar las líneas que se van dibujando en ese momento. Contrariamente a lo que se podría esperar, estas líneas no se dibujan consecutivamente. La primera línea de pixels aparece en la parte superior de la pantalla, pero la línea siguiente no aparece inmediatamente debajo, sino 8 líneas más abajo, y la tercera aparece 16 líneas más abajo, y así las demás. Cuando se llega a la línea 64, el proceso vuelve a situarse en la parte superior de la pantalla y se dibuja una segunda línea que se traslada a la novena línea. Todo ello se repite en la segunda zona de la pantalla y, por último, en la tercera. La figura muestra en un diagrama todo el proceso.

Para utilizar el archivo de visualización (abreviado, *DFile*) y calcular la dirección adonde acudir para imprimir y posicionar los pixels, obsérvese este segundo diagrama.

Byte alto
Byte bajo
0 1 0
Dirección en archivo visualización
Pixel No.
Columna No.
Línea No.

Direcciones útiles
El dibujo ilustra la importancia de los bits de una dirección dentro del archivo de visualización. Aunque parezca complicada, la estructura del archivo de visualización permite una rápida manipulación de los datos de pantalla empleando instrucciones de registros individuales, en lugar de las más profusas (y lentas) operaciones de empleo de registros pares

Aquí se muestra cómo se disponen los bits en la dirección del *DFile* con el número de pixel referido a la línea de pixel dentro de cada carácter. Así, suponiendo que estamos en el pixel superior de un carácter, el número de pixel sería el 0.

Este esquema muestra también el hecho de que si deseamos hacer bajar un pixel del carácter, todo lo que hay que hacer es incrementar el registro H (alto), o sea, INC H. Si queremos desplazar un carácter a la derecha, emplearemos HL. Para calcular la posición de una coordenada de PRINT AT se empleará la rutina del cuarto listado, la cual exige que el número de columna esté en E y el número de línea en D. Una vez llamada la rutina, la dirección se encontrará en HL.

1ª fila 1er carácter: 4000 1.º; 4100 9.º; 4200 17.º; 4300 25.º; 4400 33.º; 4500 41.º; 4600 49.º; 4700 57.º
2ª fila 1er carácter: 4020 2.º; 4120 10.º; 4220 18.º; 4320 26.º; 4420 34.º; 4520 42.º; 4620 50.º; 4720 58.º
8ª fila 1er carácter: 40E0 8.º; 42E0 16.º; 42E0 24.º; 43E0 32.º; 44E0 40.º; 45E0 48.º; 46E0 56.º; 47E0 64.º
9ª fila 1er carácter: 4800 65.º; 4900 73.º; 4A00 81.º; 4B00 89.º; 4C00 97.º; 4D00 105.º; 4E00 113.º; 4F00 121.º

32 bytes
0 7 15 23 31
4000 0 — 401F
4800 8 — 481F
5000 16 — 501F
50E0 25 — 50FF
BLOQUE 1 4000 - 47FF
BLOQUE 2 4800 - 4FFF
BLOQUE 3 5000 - 57FF

Caroline Clayton

Memoria de pantalla
Este esquema muestra la distribución del archivo de visualización del Spectrum. La pantalla se divide en tres bloques principales, cada uno de los cuales representa 256 posiciones de caracteres (8 líneas por 32 caracteres). Dentro de cada bloque las líneas de pixels se almacenan según el orden que se indica. Así el primer bloque de 256 bytes de la memoria de pantalla contiene datos para la primera línea de pixels de los primeros 256 caracteres, los siguientes 256 bytes contienen datos para la segunda línea, y así sucesivamente. Esto puede verse en el dibujo detallado de los cuatro caracteres en la izquierda de la pantalla, donde cada carácter tiene sus líneas de pixels numeradas con (1) la dirección correspondiente dentro del archivo de visualización y (2) el orden en que se rellenan las líneas de pixels cuando los datos son POKE consecutivamente dentro del archivo

```
            ORG  60000
3EAF        LD   A,175
92          SUB  D
57          LD   D,A        ;D contiene 175-coord. y
A7          AND  A
1F          RRA
37          SCF
1F          RRA
A7          AND  A
1F          RRA             ;mezcla de bits
AA          XOR  D
E6F8        AND  248
AA          XOR  D
67          LD   H,A        ;H se convierte en...
7B          LD   A,E        ;64+8*INT(B/64)+B(MOD 8)

CB07        RLC  A
CB07        RLC  A
CB07        RLC  A
AA          XOR  D
CB07        RLC  A
CB07        RLC  A;         mezcla de bits
6F          LD   L,A        ;L tiene ahora 32*INT(...
7B          LD   A,E        ;...B(MOD64)/8)+INT(x/8)
E607        AND  7          ;el byte LO es x(MOD 8)
C9          RET
```

Si deseamos calcular la dirección del pixel, utilizaremos la rutina de este quinto listado. Éste es similar al precedente, ya que también se ha de cargar D con la coordenada y, mientras que E recibirá la coordenada x. Al ser llamada esta rutina, la dirección estará en HL, que retendrá un valor entre 0 y 7, según sea el grupo de bits para el pixel adecuado.

```
            ORG  60000
7A          LD   A,D
E6F8        AND  248        ;desenmascara bits 0-2
CBF7        SET  6,A
67          LD   H,A
7A          LD   A,D
E607        AND  7          ;desenmascara bits 3-7
CB0F        RRC  A
CB0F        RRC  A
CB0F        RRC  A          ;mueve hacia arriba los
                            bits
83          ADD  A,E        ;añade bits columna
6F          LD   L,A
C9          RET
```

Un último detalle que se ha de recordar sobre el D*file* es que cada byte que examinemos (POKE) corresponde a ocho pixels. Estos pixels se almacenan del mismo modo que los almacenados en un byte de gráficos definidos por el usuario.

El archivo de atributos

La segunda parte de la memoria de pantalla (el archivo de atributos) es más inmediata que el archivo de visualización. El primer byte está en 22528 y la longitud de este archivo es de 768 bytes. Ejecute:

```
10 FOR F=22528 TO 22528+767
20 POKE F,0
30 NEXT F
```

Observará que la pantalla se llena como usted esperaba: completada la primera línea pasa al comienzo de la segunda y así hasta llenar toda la pantalla.

Cada byte en el archivo de atributos se corresponde con los valores de INK, PAPER, BRIGHT y FLASH para cada recuadro de carácter. Los tres primeros bits contienen el color de INK, y los bits 3, 4 y 5 el color de PAPER. El valor adecuado de BRIGHT está en el bit 6 y el de FLASH en el bit 7.

Una observación final sobre las alteraciones de color de BORDER. Para cambiar de color el recuadro desde código máquina, A se cargará con el número de color y se llamará #229BH.

Variables del Spectrum

Nombre variable	Posición	Descripción
COORDS	23677	Contiene la coord. x del último punto trazado
COORDS	23678	Contiene la coord. y del último punto trazado
DFCC	23684	Es una variable de dos bytes que contiene la dirección de memoria en el archivo de visualización del primer byte que se empleará para imprimir
S POSN	23688	Contiene la posición actual de impresión para columnas
S POSN	23689	Contiene núm. de línea actual utilizado para impresión
ATTR P	23693	Se usa para almacenar los colores permanentes establecidos por INK, PAPER, FLASH y BRIGHT. Los datos están en el formato ATTR
MASK P	23694	Se usa para desenmascarar ATTR P cuando se emplean colores transparentes (INK9, etc.). Cada bit encendido en el correspondiente bit de ATTR P será ignorado
ATTR T	23695	Como ATTR P, pero sólo se usa para colores temporales; p. ej., para cambios de color en una instrucción PRINT
MASK T	23696	Igual que MASK P, pero para colores temporales
P FLAG	23697	Varios flags empleados para almacenar el estado tanto temporal como permanente. Cada bit muestra si lo que ha de ser transparente es la tinta o el papel

N.B. Ya fueron descritas precedentemente ATTR P, MASK P y P FLAG

Rutinas ROM del Spectrum

Nombre rutina	Posición	Requisitos entrada	Resultado
RST # 10	#0010	A debe contener el código ASCII del carácter a imprimir	Se imprimirá el carácter
PR STRING	#203C	BC debe contener la longitud de la cadena; DE contendrá la posición de la cadena en la memoria	La cadena se imprimirá en la posición sig. disponible en pant.
PRINTED FP	#2DE3	El número a imprimir debe estar encima de la pila de cálculo	Se imprime el núm. sup. en la pila calculadora
STK-A	#2D28	A contendrá el valor a colocar en la pila	El valor en A se sitúa encima de la pila calculadora
STK-BC	#2D2B	BC debe contener el valor a colocar en la pila	El valor en BC se sitúa encima de la pila calculadora
PLOT	#22E5	B y C contendrán respectivamente las coordenadas x e y	Se traza el punto (x, y) y se actualizan las COORD SV
DRAW	#24BA	B contendrá ABS(y) C contendrá ABS(x) D contendrá SGN(y) E contendrá SGN(x)	Si los valores son mayores que 1 la línea será de puntos
CHAN-OPEN	#1601	A debe contener el número del flujo que se va a usar	Abre el canal adecuado


E R T Y U I O
D F G H J K L
X C V B N M

9 788485 822836
10100